Universidad
Carlos III de Madrid

Grado en Ingeniería Informática

Trabajo de Fin de Grado

Desarrollo de una aplicación multimodal para apoyar el estudio mediante m-learning

Autor: Cristhian Edmundo Villegas Flores

Tutor: Dr. David Griol Barres

Madrid, marzo de 2017

Resumen

En el presente Trabajo de Fin de Grado se ha desarrollado una aplicación para Android en el campo del *m-learning* que consiste en un sistema cuyo objetivo es complementar el estudio de los alumnos mediante la realización de preguntas tipo test.

La aplicación permite que el usuario se inscriba en las asignaturas en la que se ha matriculado y que está cursando en la universidad, instituto, o colegio. De esta manera puede acceder al contenido relacionado con dichas asignaturas a modo de repaso o refuerzo después de haber estudiado el temario correspondiente.

Al principio del documento se plantea el problema, la motivación que ha llevado a realizar este proyecto y se marcan unos objetivos que se tienen que cumplir a medida que se avanza el proyecto.

A continuación, se realiza un estudio acerca de los distintos sistemas operativos para teléfonos móviles que ofrece el mercado y por qué se ha elegido Android sobre el resto. Asimismo, analizan las diferentes tecnologías y herramientas software disponibles en el mercado actual que mejor se adecúen al sistema planteado y que mejores prestaciones ofrezcan.

En la parte más extensa de este documento se explica detalladamente cómo se ha desarrollado la aplicación. Se describe el diseño de la interfaz de usuario de la aplicación, la funcionalidad de cada uno de los módulos y la comunicación e intercambio de datos que se realiza con la parte del servidor.

Por último, se especifican las pruebas realizadas sobre la aplicación y se comprueba que satisfacen los requisitos planteados para el correcto funcionamiento del sistema completo, así como de cada uno de sus módulos por separado.

Abstract

The current paper corresponds to the document of the bachelor thesis which title is “Developing a multimodal application to support the study through m-learning”. The project’s main target is the develoment of an Android application, named TestMe, that helps the students to prepare their exams by performing multiple choice questions.

Introduction

The main motivation to perform this project is the huge increase of smartphone users along the last years. For instance, in Spain, the 88% of people over 15 years old owns a smartphone with internet connection.

The second motivation, a major reason why the students get bad scores in their exams is because either they don’t study properly or they don’t study enough time. Therefore, the system to be created is aimed to support and help the students to prepare their exams.

Likewise, Spain has the highest school dropout rate in the European Union, where almost the 20% of the students left their studies, in 2015. Given the information about the rate of young people who owns a smartphone, the idea is to use this application to help young people by using technology, they are familiar with, to study in a more enjoyable way.

Goals

As said before, the project’s main goal is the develoment of an Android application focused on the educational field, m-learning, which main purpose is to help, reinforce and support students when preparing their exams.

The application will help the user to determine the knowledge that he or she has acquired, of a certain subject, by answering a round of questions. In these rounds ten multiples choice questions are proposed which the user can answer in a multimodal way, i.e., the user can answer either through oral interaction or through the use of the touch screen.

Depending on the scores obtained the user can know which subjects he/she has prepared in an appropriate way and which subjects need more attention. In this way, the user complements his/her learning of the subjects and the preparation of the exams.

In order to achieve the proposed goals, the following points must be accomplished.

- To enlarge the knowledge of the Android platform and its application development environment.
- To perform an analysis of the technologies and tools that are used to run the application, evaluating the possible options to implement a client-server architecture.
- To present examples of alternatives and existing applications that are similar to the application that is going to be developed in this project.
- To perform an analysis of the different possibilities for the development of applications that provide speech recognition.

Structure

The paper follows the structure described in the next points. These points contain a brief introduction for each section of the paper as well.

1. *Chapter 1: Introduction*. The paper begins with the first chapter which is dedicated to introduce the document's subject. This first chapter shows the motivation that led to perform this whole project and the project's goals.
2. *Chapter 2: State of the art*. This second chapter brings a study of the different operating systems and the different current technologies for mobile devices. It explains the decision to select Android as the platform for the implementation of the application as well.
3. *Chapter 3: System description*. The third chapter analyzes the different technologies and software tools and the languages used to develop the whole system.
4. *Chapter 4: Application design and development*. In the fourth chapter the design of the user interface is described. For each of the components its functionality and data flow is detailed.

5. *Chapter 5: Evaluation of the application*. The fifth chapter details the tests that have been performed on the system to ensure that it works properly and to evaluate whether the proposed objectives have been achieved.
6. *Chapter 6: Project Management*. The sixth chapter describes the Project schedule and the project's budget.
7. *Chapter 7: Conclusions and future work.* The seventh chapter describes the problems found, proposes possible improvements in the developed system and possible future functionalities.

Summary

Android is the chosen operating system (OS) to develop the application. The reasons to choose this platform are the number of users and devices using this OS all over the world, the application development costs and the application publishing costs.

It has taken into account the number of mobile devices sold all over the world when choosing the operating system, because the greater the number of smartphones running the operating system, the greater the number of users the application can reach.

- Android continues to be the dominant platform, leading most markets with 1240 million devices sold in 2016.
- Android is positioned as the best option, since it has a market share of 85.2% of devices sold.

It has taken into account the costs of development of an Android application and the costs of publishing an Android application as well.

- Android applications are developed with the Android Studio software which is free to use and it is available for Windows, Mac and Linux.
- Google makes a one-time charge of $25 to get a developer account on Google Play, which allows you to publish apps.

Therefore, Android is the best choice since it is the most used operating system. Plus, the cost of developing and publishing an application is a one-time payment of $25.

System Description

The system counts with a client-server architecture. More specifically, both client side and server side are composed of different subsystems that interact and cooperate with each other making the communication between the client side and server side possible.

- The client side consists of the Android application.
- The server side contains the web server and the database.

The Android application –client– requests an action on the database, for that it sends a HTTP request to the web server.

The PHP files make up a middleware between the application and the database. This PHP middleware, i.e, the web service takes the HTTP request and turns it into a SQL query which is sent to the database.

The database processes the SQL query, then it creates, retrieves, stores or deletes the requested data. Afterwards, the database returns the results to the web server.

The web server receives the data returned by the database and converts this data into JSON format and sends it to the client application.

The application obtains the data in JSON format, then the app analyzes the data and performs corresponding actions.

Multiple software has been used to develop both sides, client and server, of the system and to implement the functionality explained before.

Android Studio has been used to implement the code, the functionality and to design the user interface (all the java and xml files) of the application, i.e., the client side. The reason to choose Android Studio is that it is the official IDE (Integrated Development Environment) of Google and it is software specifically created to develop Android applications. The minimum version requested for the application is Android 4.4 KitKat. In addition, the target devices for the app are smartphones and tablets.

To implement the server side the software used were Hostinger, MySQL, phpMyAdmin, Filezilla and Postman.

Hostinger is a website dedicated to web hosting. It has been used to host the web server, the database and their content. Hostinger offers a free plan for students that's why it has been chosen among all the many possibilities. The free plan gives enough features to develop the system. For instance, Hostinger offers 2000 MB of disk space, 100 GB of bandwidth, PHP support, FTP access and two MySQL databases.

MySQL is an open-source relational database management system. It has been used to store all the data related to the application's users, the degrees, the subjects, the topics, the questions and all the other data related to the application in the database. Plus, MySQL is the default database manager system provided by Hostinger.

PhpMyAdmin is a free and open-source software tool which main task is to handle MySQL systems. It has been used to to manage the database.

Filezilla is a free software and multiplatform FTP (file transfer protocol) application and it has been used to upload the php files, which perform the web service, to the web server hosted on Hostinger.

Finally, Postman is a open-source software tool that allows to easily manage requests to REST services. Postman has been used to test the correct performance of the web service.

System design and development

Again, it is necessary to differentiate between client side and server side in order to explain how the design of the system was made and how it was developed.

In one hand, we have the client side. As said before, the client side corresponds to the application, so in order to explain the application design and development is necessary to explain the user interface and its functionality.

The main features that the application's interface must fulfill are described in the following points:

- Simplicity: The interface must be intuitive and minimalist to simplify its use.
- Clarity: The interface must be perfectly understandable to the users.
- Familiarity: Concepts well known for the users must be used.
- Consistency: Similar actions and concepts must have similar representation and layout in the interface.
- Reliability: The behavior of the interface must always be as expected, so that it does not cause loss of information or malfunction of the application.
- Easy to Access: All content must be accessible without having to go through complicated or long paths.

The user interface of the app is composed of eleven screens, each of these screens have their own purpose on the application's behavior. These are the screens of the applications with a brief explanation of their functionality:

1. Login screen: This screen allows the user to log in the application by identifying himself/herself with his/her username and password, which must have been previously registered. This screen also allows the user either to access the sign up screen if he/she has not registered in the application before or to access to the password recovery screen.
2. Sign up screen: This screen allows the user to register his/her username and password so he/she can access the contents of the application later. This screen also allows the user to register the user's email address and bachelor degree. Afterwards the application sends the data to the database so the data becomes persistent.

3. Password recovery screen: In this screen, the user can retrieve the access password to the application via an email sent to the email address entered by the user.
4. Subject enrollment screen: In this screen the user chooses the subjects, by marking the corresponding checkboxes, that he/she wants to enroll and study. All the subjects shown are related to the bachelor degree registered by the user.
5. Home screen: This one is the main screen of the application. This screen show a list with the subjects enrolled by the user. The user can choose one of these subjects to access its content, i.e., its syllabus.
6. Add/Remove subjects screen: This module is composed of two screens, actually. The first one shows a list with all the subjects that have not been enrolled by the user so the user can add a new subject. The other one shows a list with the already enrolled subjects so the user can remove one subject when he/she has finished studying that subject.
7. Topic list screen: This screen show a list with the topics belonging to a subject selected by the user. The user can choose one of these topics to access its content, i.e., its questions.
8. Question screen: This screen shows a question, related to the subject and topic selected, and its four possible answers. The user has to choose one of the four options, where only one of them is correct. The user can give the answer by using the touch screen or through the speech recognition. Once the user has provided an answer the system determines whether it is correct or not.
9. Result screen: This screen appears after the round of ten questions. Its purpose is to present the results obtained in the last round of questions performed. So the user will know if it is necessary to study that subject more.
10. Statistics screen: This module is composed of two screens. The first one shows a list with the number of exams performed and the average score for all the subjects enrolled by the user. The other one shows a list with the number of exams performed and the average score for all the topics belonging to a specific subject. The main purpose of these two screens is to show the user the global and specific progress followed by him/her for each one of the subjects.
11. Profile screen: This screen shows the following information related to the user: email address, username, password (hidden) and bachelor degree. All the data can be edited except the email address.

In the other hand, there is the server side. The server side is composed of the web server and the database.

The web server, contracted via Hostinger, hosts the php files. These php files altogether work as a web service. The web service behaves as a middleware between the application and the database. Each one of the php files has its own purpose, for instance, to connect the application with the web server, to start a connection with the database, to retrieve, to create, to insert, to update and to delete data from the database.

The database contains several tables where all the data related to the application are stored. These tables of the database store data related to subjects such as subejct's name, which degree the subject belongs and which course the subject belongs; related to topics such as topic's name, topic's number and which subject it belongs; related to questions such as the question itself, the four answers, the correct answer as a number, and which topic and subject the question belongs. The database stores the information about the user as well, information as username, email address, password, scores, statistics, etc.

Whenever the application, while is being used by the user, performs an action that includes the database data, like signing up or answering questions, the web service is required. The application sends a request to perform a determined action on the database, then the web service takes care of accomplish the request succesfully. Aftewards the web service sends a response to the application in order to determine wether the action was accomplished or not. All this process means that the application is connected to the database trhough the web service, i.e., the web service works as a middleware.

Results

At this point, the application has been already developed. Thus, several testing has been applied to the system to prove the correct performance of its modules and functionalities.

First of all, unit testing have been performed to check and verify the correct behavior of each of the system's modules. For instance, it has been tested that a username that is already in use can not be registered again; it has been tested that while a user is trying to log in the application, the username and password he/she entered must match with the username and password that he/she registered before; it has been tested that speech recognition work properly, so the user can answer by tapping the screen or by speaking, etc.

The database has been tested using phpMyAdmin. This software allows to perform sql queries, this way we can test if the data retrieved is correct or if the data has been inserted correctly or if the tables are correctly related.

The web service has been tested using Postman. Postman is a software that allows to manage REST requests. Thus, every php file added into the server has been tested with the correspondent request in order to see its behavior, so we can know if the response is correct or not.

After proving that the modules work properly separately, several tests have been performed in order to check and verify that the system's modules behavior is correct altogether. It has been tested that the user can sign up in the application and afterwards he/she can log in the application by entering these data. It has been tested that the application allows the user to enroll new subjects. It has been tested that the application allows the user to remove subjects previously enrolled. It has been tested that the system counts the number of correct and wrong answers throughout a round of questions. It has been tested that the system allows the user to edit his/her data, so if a user edit his/her username or password the application must grant access with the new data and deny it with the old data.

The application has passed all the tests that have been carried out. Therefore, the system provides a set of basic and essential functionalities that make the application really helpful to the user. So there can be a progress in the user's study and subsequently an improvement in his/her marks.

Conclusions

In relation to the objectives established before starting the project, most of them have been successfully achieved.

In order to accomplish these goals, I have worked with some technologies that I barely used few times before and some other technologies that I had not previously worked with. It is necessary to comment that it has been very useful to deepen the knowledge about these technologies or to learn to use them, not only for the development of this bachelor's degree project but for future projects too.

Among these technologies we can highlight the web services learnt and used in a couple of subjects of the career, but not very deeply; PHP, a very simple language to program the server side, which has been learnt especifycally for this project; and the Android's library Volley which is used to make connections over the internet in a very simple way.

Finally, this work has been a big personal and a professional challenge. Many hours of work have been dedicated to the accomplishment of this project; to collect the information and research; to develop the application; to write this report, etc. Furthermore, the realization of this project has given me a new approach about the knowledge acquired along the career and it has shown me where to lead my laboral future.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

1. Introducción

Este capítulo está dedicado a introducir el Trabajo de Fin de Grado, indicando la motivación del mismo y los objetivos propuestos, se define la estructura que sigue este documento y, por último, los recursos que son necesarios para la realización del trabajo.

1.1 Motivación

La primera motivación para realizar este proyecto es el gran aumento de usuarios de *smartphones* (teléfonos inteligentes) durante los últimos años. La expansión de los dispositivos telefónicos ha causado gran impacto en la sociedad, de tal manera que un gran porcentaje de la población posee uno, hay aproximadamente más de 2,08 mil millones de usuarios de *smartphones* en el mundo. En España, un 98% de los jóvenes entre 10 y 14 años y un 88% de los mayores de 15 años posee un teléfono de última generación con conexión a internet [1].

La segunda motivación es porque dentro del ámbito de estudios superiores, los estudiantes suelen obtener malos resultados, malas notas en los exámenes y hasta suspender asignaturas debido al estudio insuficiente o incorrecto de la materia. Por ello, se ha decidido elaborar un sistema de apoyo y refuerzo que ayude a los estudiantes a preparar sus exámenes.

De la misma manera, teniendo presente que España tiene el índice más alto de abandono escolar en la Unión Europea, donde casi un 20% de los alumnos abandonaron sus estudios en 2015 y uno de cada cinco estudiantes deja de formarse tras acabar la ESO [2]. Y dados los datos sobre el porcentaje de jóvenes españoles con acceso a teléfonos inteligentes, utilizar el sistema que se va a introducir en este documento para ayudar a los jóvenes mediante el uso de tecnología, con la que están familiarizados, a estudiar de una manera más amena.

1.2 Objetivos

El principal objetivo del presente Trabajo de Fin de Grado es el desarrollo de una aplicación para Android enfocada al ámbito educativo, *m-learning*, cuyo propósito primordial es ayudar, reforzar y apoyar a los estudiantes universitarios a lo hora de preparar sus exámenes.

La aplicación ayudará al usuario a determinar los conocimientos que ha adquirido, de una asignatura determinada, mediante la realización de simulacros de exámenes. En dichas pruebas se proponen diez preguntas de tipo test que el usuario puede contestar de manera multimodal: mediante interacción oral o mediante el uso de la pantalla táctil.

Dependiendo de los resultados obtenidos el usuario puede saber qué asignaturas ha preparado de una manera adecuada y qué asignaturas necesitan más atención. De esta manera, el usuario complementa el aprendizaje de los temas y la preparación de los exámenes.

Para conseguir el objetivo planteado, se debe cumplir los siguientes puntos:

- Profundizar los conocimientos de la plataforma Android y de su entorno de desarrollo de aplicaciones.
- Realizar un análisis de los sistemas de diálogo hablado para desarrollar aplicaciones que proporcionen reconocimiento del habla que ofrece Android.
- Realizar un análisis de las tecnologías y herramientas que se van a emplear a la hora de desarrollar la aplicación, valorando las distintas posibilidades para implementar una arquitectura cliente-servidor.
- Presentar ejemplos de aplicaciones alternativas ya existentes que sean similares a la aplicación que se va a desarrollar en este proyecto.

1.3 Estructura del documento

En este apartado se define la estructura del presente documento. Además, se describe la información que contiene cada sección del documento.

El primer capítulo, *Capítulo 1: Introducción,* contiene una introducción al proyecto, la motivación que ha llevado a realizarlo, los objetivos propuestos, la estructura del documento y los materiales empleados.

En el segundo capítulo, *Capítulo 2: Estado del arte,* se realiza un estudio de los diferentes sistemas operativos para dispositivos móviles y las distintas tecnologías actuales. Asimismo, se explica la decisión de seleccionar Android como plataforma para la realización de la aplicación.

En el tercer capítulo, *Capítulo 3: Descripción del sistema,* se analizan las distintas tecnologías y herramientas software y los lenguajes empleados utilizadas para desarrollar el proyecto.

En el cuarto capítulo, *Capítulo 4: Diseño y desarrollo de la aplicación,* se describe la interfaz de usuario y el servicio web. Para cada uno de los componentes de la interfaz se detalla su diseño, funcionalidad y su flujo de datos.

En el quinto capítulo, *Capítulo 5: Evaluación de la aplicación,* se detallan las pruebas que se han realizado sobre el sistema para garantizar que funciona correctamente y para evaluar si se han alcanzado los objetivos propuestos.

En el sexto capítulo, *Capítulo 6: Gestión del proyecto*, se incluye la planificación del proyecto, el resumen de costes que ha conllevado el desarrollo del sistema completo, el entorno socioeconómico y el marco regulatorio.

El séptimo capítulo, *Capítulo 7: Conclusiones y trabajo futuro,* describe los problemas encontrados, plantea posibles mejoras en el sistema desarrollado y posibles funcionalidades futuras.

1.4 Recursos

En este apartado se realiza una lista con los materiales y recursos, hardware y software, que son necesarios para la realización de este proyecto.

Recursos hardware

- Ordenador portátil: MacBook Pro 13”.
- Smartphone: Samsung Galaxy S5.

Recursos software

- Android Studio IDE: entorno de desarrollo para aplicaciones Android.
- Filezilla: aplicación que permite transferir archivos a un servidor.
- Gantt Project: herramienta que permite realizar planificación de tareas.
- JDK (*Java development kit*): kit de desarrollo de software de Java.
- Microsoft Office 365.
- Motor de síntesis de voz de Google.
- PhpMyAdmin: herramienta que permite administrar bases de datos.
- Postman: herramienta que permite gestionar peticiones a servicios web.
- Volley: biblioteca que facilita la creación de conexiones para aplicaciones Android.

Capítulo 2

2. Estado del arte

Este capítulo está dedicado al análisis del entorno actual de las tecnologías móviles y su evolución durante los últimos años. Se realiza, también un análisis del estado del arte relativo a la resolución de los objetivos propuestos en el apartado 1.2.

A lo largo del capítulo se realiza un estudio de cada uno de los sistemas operativos indicando sus puntos fuertes y sus desventajas con respecto a los demás. Asimismo, se justifica la elección de Android como sistema operativo para la aplicación.

De la misma manera, se realiza un estudio sobre los sistemas de reconocimiento automático del habla (*Automatic Speech Recognizer*).

Por último, se presentan programas o aplicaciones ya existentes similares al sistema que se va a desarrollar.

2.1 Dispositivos móviles

Los dispositivos móviles son computadores lo suficientemente pequeños y ligeros como para ser transportados por una persona, y que disponen de la capacidad de batería suficiente para funcionar de manera autónoma [3].

Los siguientes dispositivos móviles son los más utilizados hoy en día:

- *Smartphone*: También llamados teléfonos inteligentes, son dispositivos móviles pequeños que poseen una pantalla táctil. Sus principales funciones son realizar llamadas, uso de aplicaciones y acceso a internet. El 87% de las líneas móviles corresponden a *smartphones*. El 90% de los usuarios de *smartphones* se conecta todos los días a internet en España [4].
- *Tablet*: También llamados tabletas, son dispositivos móviles de mediano tamaño y poco peso, ligeramente mayores a los *smartphones*. Sus principales funciones son el uso de aplicaciones y acceso a internet. El 38% de la población española tiene alguno en casa [5].

2.1.1 Sistemas operativos móviles

Un sistema operativo es un sistema software diseñado para administrar y controlar el hardware de un computador. Los sistemas operativos que funcionan en dispositivos móviles son los sistemas operativos móviles.

Las ventas mundiales de dispositivos móviles, especialmente de *smartphones*, ha crecido durante los últimos años. En 2015, se vendieron 1400 millones de unidades de *smartphones* en todo el mundo, un incremento del 14.4% con respecto a 2014 [6].

Los próximos puntos dan una visión general de los diferentes sistemas operativos móviles más utilizados.

2.1.1.1 Android

Android es un sistema operativo para dispositivos móviles de código abierto desarrollado por *Google*. La arquitectura del sistema está basada en el *kernel* de Linux y Java. Esta combinación trae consigo algunas características de seguridad, gestión eficiente de memoria, multitarea apropiativa, identificación de usuarios Unix (*UIDs*), permiso de archivos y el tipado seguro (*type safe*) de Java.

Android se ha expandido más allá de los *smartphones* y *tablets*, siendo utilizado en televisiones mediante *Android TV*, en coches mediante *Android Auto* y *smartwatches* mediante *Android Wear* [7].

Cada versión de Android tiene asociado el nombre de un dulce que además sigue orden alfabético [8]. A continuación, se describen las versiones que ha tenido Android a lo largo de su existencia [9].

Android Apple Pie

La primera versión oficial y definitiva, encargada de estrenar la plataforma en el mercado. *Android 1.0*, llamado posteriormente *Android Apple pie*, ofrecía características y funciones básicas. Entre ellas destacan *Android Market* (tienda de aplicaciones), compatibilidad y sincronización con los diferentes servicios de Google: Maps, Contactos, Calendar o Youtube.

Android Banana Bread

Esta versión fue lanzada para corregir problemas de la versión 1.0. Introdujo mejoras e incorporó ciertas funcionalidades como la posibilidad de adjuntar archivos multimedia a los mensajes o información de usuario en Google Maps.

Android Cupcake

El lanzamiento de esta versión introdujo nuevas características para los usuarios y los desarrolladores:

- Se incluyeron los *widgets* en las pantallas de Android, que son las vistas en miniatura de las aplicaciones.
- Reproducción de formatos 3GP y MPEG4.
- Posibilidad de copiar y pegar texto.
- Soporte para *Bluetooth* y para teclados virtuales de terceros con autocorrectores y diccionario para palabras personalizadas.
- Transacciones animadas.
- Reconocimiento de voz.
- Mejoras en las aplicaciones integradas del sistema, como los contactos o *Youtube*.
- Mejoras en la interfaz de usuario.

Android Donut

Esta versión fue lanzada en 2009 para una gran variedad de dispositivos. Introdujo las siguientes novedades:

- Motor *text-to-speech*, es decir, síntesis de habla que permite convertir texto a voz.
- Mejora de velocidad en búsqueda y aplicaciones de cámara.
- Mejora de la integración de la galería con la cámara y la videocámara.

Android Eclair

Esta versión vino acompañada de una gran lista de mejoras y novedades, entre las que se incluyen soporte de múltiples cuentas de correo electrónico, navegador actualizado que permite el zoom mediante doble-toque y da soporte a HTML5; y gestión de cámara mejorando el enfoque, balance de blancos, zoom y flash.

Android Froyo

Froyo es una de las versiones más conocidas, fue lanzada en 2010 ofreciendo una amplia lista de novedades:

- Posibilidad de instalar aplicaciones en la memoria externa.
- Soporte para pantallas de mayor resolución.
- Optimizaciones de velocidad, memoria y rendimiento.
- Dictado de voz mediante *Bluetooth*.
- Soporte para Adobe Flash.
- Añade las funcionalidades *Wi-Fi hotspot* y *USB tethering* [10].

Android Gingerbread

Esta versión actualizó el diseño de la interfaz de usuario y realizo incrementos en la velocidad y simpleza. Además, se añadieron las siguientes características:

- Soporte para *Near Field Communication* (NFC).
- Soporte para *Google Wallet*.
- Soporte para chat de voz y vídeo mediante *Google Talk*.
- Soporte nativo para VoIP, es decir, telefonía por internet.
- Soporte para múltiples cámaras, se empezaron a incluir cámaras frontales a los dispositivos.
- Se da a los usuarios la posibilidad de acceder a los archivos descargados mediante un nuevo gestor de descargas.
- Mejora en la eficiencia de la batería.

Android Honeycomb

Esta versión fue diseñada para funcionar en pantallas grandes como los *tablet*. Es exclusiva para estos dispositivos, por lo que no estuvo disponible para *smartphones*.

La principal característica introducida en esta versión es la mejora de la interfaz de usuario para su uso en *tablets*, se rediseñaron los *widgets*, el teclado, la barra de notificaciones. Todos estos elementos ajustados al tamaño de pantallas mayores.

Android Ice cream Sandwich

En la versión *Ice Cream Sandwich* se llevó a cabo un rediseño de la interfaz de usuario, se añadieron botones nuevos, barra de notificaciones, iconos. Asimismo, se incluyó la grabación de vídeos de alta definición 1080p y soporte de Wi-Fi Direct.

Otras novedades añadidas fueron el desbloqueo del teléfono mediante reconocimiento facial, nueva tipografía, *Roboto*; y captura de pantalla integrada.

Android Jelly Bean

Jelly Bean fue la primera versión de Android para *smartphones* y *tablets* simultáneamente, se conservó la estética presentada en *Ice Cream Sandwich*. Sin embargo, se añadieron novedades que mejoraron la fluidez general del sistema.

Además, se mejoró la búsqueda por voz de Google, permitiendo realizar búsquedas pronunciando “Ok, Google”. De esta manera, dejó de ser necesario utilizar el icono del micrófono.

Android Kitkat

En la versión *Kitkat* se arreglaron y mejoraron funcionalidades y características introducidas en versiones anteriores.

Se rediseñaron elementos de la interfaz como la barra de notificaciones, los menús o los iconos, todos fueron pequeños cambios que reafirmaban el concepto de interfaz introducido en *Ice Cream Sandwich*.

Android Lollipop

En *Lollipop* se desarrolló un nuevo patrón de diseño llamado *Material Design*. La nueva interfaz gráfica tiene un aspecto minimalista y colorido.

Se realizaron mejoras en el ahorro de la batería y se añadieron nuevas maneras de controlar las notificaciones, pudiendo verlas y contestarlas desde la pantalla de bloqueo. Además, se dio la opción de personalizar las prioridades para las notificaciones.

Se dio soporte al uso múltiple de tarjetas SIM y se añadió protección a los dispositivos para que en caso de pérdida o robo éstos permanecieran bloqueados.

Android Marshmallow

En *Marshmallow* se realizaron pocos cambios en la interfaz respecto a *Lollipop*, manteniendo *Material Design* como base para el desarrollo de estos cambios. Las novedades más importantes introducidas en esta versión fueron:

- *Google Pay*, la plataforma de pagos de *Google*.
- Soporte nativo para los sensores de huellas dactilares.
- *Now on Tap*, se expande *Google Now* a todo el dispositivo.
- Soporte completo para USB y tarjetas SD.
- Administrador de permisos, concede a los usuarios la posibilidad de decidir los permisos a los que puede acceder cada aplicación.
- Compatibilidad con lápices *Bluetooth*.
- Copias de seguridad de datos automáticas y completas de las aplicaciones.
- Se renueva la pantalla de bloqueo.
- Configuración de la interfaz de usuario del sistema, esto permite personalizar los iconos que aparecen en la barra de estado, o los ajustes rápidos.

Android Nougat

Nougat, es la versión 7.0 de Android. Fue oficialmente estrenada el 22 de agosto de 2016.

Nougat presenta cambios notables en el sistema operativo. Quizá el cambio más notable es el modo de pantalla dividida, en el que dos aplicaciones se presentan a la vez, ocupando cada una la mitad de la pantalla. A la funcionalidad multiventana se añade la posibilidad de arrastrar contenido de una aplicación a otra.

Se da la posibilidad de editar los accesos directos que se muestran al deslizar la barra de notificaciones hacia abajo. También destacan las notificaciones agrupadas, en el que se agrupan notificaciones, que pertenecen a una misma aplicación, como si fuese solo una. Para ver las notificaciones con más detalle se pueden mostrar por separado.

Otra novedad relacionada con las notificaciones es la posibilidad de responder a mensajes directamente desde la notificación, sin necesidad de acceder a la aplicación.

En la Figura 1 se puede ver el diseño de la interfaz presente en *Android Nougat*.

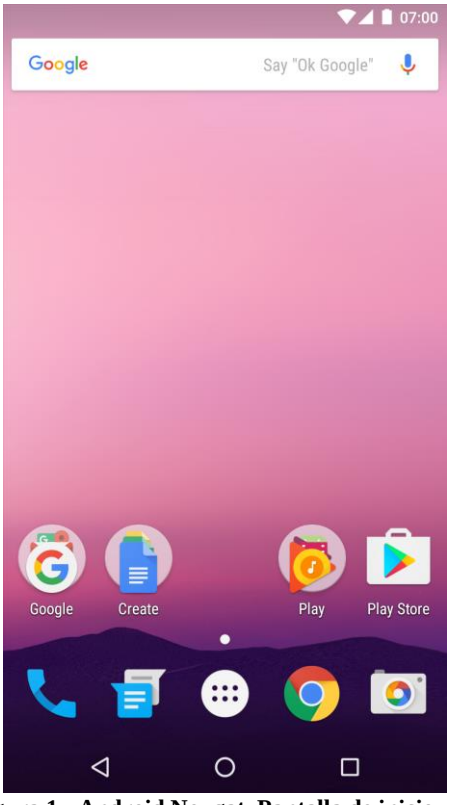

Figura 1 - Android Nougat. Pantalla de inicio.

2.1.1.2 iOS

iOS, originalmente llamado *iPhone OS*, es un sistema operativo para dispositivos móviles desarrollado y distribuido por *Apple Inc*. Fue estrenado en 2007, integrado en el *iPhone* y *iPod Touch*. Más tarde, en 2010, fue incluido también en el *iPad*.

Desde su inicio *iOS* ha ido creciendo y extendiéndose para dar soporte a cada vez más dispositivos de *Apple* como la línea completa de *iPhone* (desde el *iPhone* original hasta el recién estrenado *iPhone 7*), los diferentes *iPad* (*iPad*, *iPad Air*, *iPad Mini* y *iPad Pro)*, así como la gama completa de *iPod* (*Classic*, *Mini*, *Nano*, *Shuffle* y *Touch*), los diferentes *Apple TV* y el *Apple Watch*. Apple no proporciona licencias para instalar iOS en hardware de terceros.

La interfaz de usuario de iOS está basada en el concepto de manipulación directa, utilizando gestos *multi-touch*. iOs deriva de *Mac OS X*, por lo tanto, es un sistema operativo tipo Unix.

iOS tiene cuatro capas de abstracción: la capa del núcleo del sistema operativo, la capa de servicios del núcleo, la capa de medios y la capa *Cocoa Touch* [11]. La última versión estable de iOS, en el momento de redactar la memoria, es iOS 9.3.5.

A continuación, se realiza un resumen de las versiones que ha tenido iOS a lo largo de su existencia [12].

iPhone OS 1

La primera versión de *iOS*, llamada *iPhone Os*, fue publicada en 2007 junto con el primer *iPhone*. Esta primera versión ofrecía características y funciones básicas, no tenía *App Store*, no daba soporte 3G, ni soporte multitarea, no aceptaba aplicaciones de terceros, tampoco dejaba copiar y pegar texto.

Sin embargo, el diseño de la interfaz junto con la pantalla táctil capacitiva del *iPhone*, que reconocía gestos y toques realizados con las manos, produjeron un sistema intuitivo que se diferenciaba de la competencia, que utilizaban pantallas resistivas, en la manera de interactuar con el sistema.

iPhone OS 2

La segunda versión, *iPhone OS 2*, fue publicada en 2008 junto con el *iPhone 3G*. Esta versión introdujo la *App Store*, la aplicación de contactos y permitió el uso de aplicaciones de terceros.

iPhone OS 3

Esta versión llamada *iPhone OS 3*, fue publicada en 2009 junto con el *iPhone 3GS*. Esta versión trajo consigo una cantidad importante de novedades como la función de copiar, cortar y pegar; la búsqueda *Spotlight*, control por voz, soporte para mensajes MMS, grabación de vídeo y modo *landscape* (pantalla horizontal)

iOS 4

Esta versión llamada *iOS 4*, fue publicada en 2010 junto con el *iPhone 4*. Esta versión presumía de presentar más de 100 novedades de los cuales destacan FaceTime para realizar videollamadas, la organización de las aplicaciones por carpetas, soporte nativo para la pantalla retina del *iPhone 4*, posibilidad de tener abiertas múltiples aplicaciones a la vez (multitarea) y corrector ortográfico.

iOS 5

La versión 5.0 del sistema operativo de *Apple* fue publicada en 2011 junto con el *iPhone 4S*.

iOS 5 introdujo importantes características nuevas como el asistente virtual *Siri*, el centro de notificaciones, la aplicación de mensajería *iMessage*, sincronización con *iTunes* mediante *Wi-Fi* y *iCloud*.

iOS 6

La versión 6.0 del sistema operativo de *Apple* fue publicada en 2012 junto con el *iPhone 5*.

iOS 6 trajo nuevas mejoras y algunas novedades. La principal novedad fue *Apple Maps*, alternativa a los mapas de *Google*. Se realizaron mejoras en Siri y se integraron *Twitter* y *Facebook* como aplicaciones nativas.

iOS 7

La versión 7.0 del sistema operativo de *Apple* fue publicada en 2013 junto con el *iPhone 5S* y *iPhone 5C*.

iOS 7 presentó un diseño renovado de la interfaz de usuario, se rediseñaron los iconos con un formato minimalista, simplificado y plano, con una paleta de colores nueva y fondos animados.

Una de las principales novedades fue *Control Center*, que permite el acceso a los ajustes rápidos. Se accede a él deslizando el dedo desde la parte superior de la pantalla. Además, se realizaron mejoras en la multitarea, *Safari* y *Siri*.

iOS 8

La versión 8.0 del sistema operativo de *Apple* fue publicada en 2014 junto con el *iPhone 6* y *iPhone 6 Plus*.

Esta versión continuó la línea que marcó *iOS 7* con el diseño de la interfaz de usuario.

Con *iOS 8* llegaron los teclados de terceros, monitorización de la batería y soporte para vídeos en cámara lenta. Se añadió la posibilidad de enviar audio y vídeo mediante *iMessage*. Asimismo, se incluyeron *widgets* en el centro de notificaciones.

iOS 9

La versión 9.0 del sistema operativo de *Apple* fue publicada en 2015 junto con el *iPhone 6S* y *iPhone 6S Plus*.

Con *iOS 9* se mejoraron las prestaciones de *Siri* integrándolo con *Spotlight.* Se añadió la aplicación *Wallet* para realizar pagos, soporte para *CarPlay* inalámbrico y *FindMyiPhone* como aplicación del sistema.

iOS10

La versión 10 del sistema operativo de *Apple* fue publicada en 2016 junto con el *iPhone 7* y *iPhone 7 Plus*.

iOS 10 es la versión más reciente en el momento de redactar este documento, fue presentada pocos meses antes de entregar la memoria.

Debido a su reciente presentación se conocen pocas características nuevas de *iOS 10*. Algunas de las novedades que trae son las notificaciones interactivas, Siri funciona ahora con aplicaciones de terceros como *Whatsapp*, *Pinterest* y *Uber*.

Otra novedad importante es la posibilidad de desbloquear el terminal levantándolo, eliminando así la necesidad de deslizar el dedo por la pantalla [13].

En la Figura 2 se puede ver el diseño de la pantalla de inicio que se presenta en *iOS 10*.

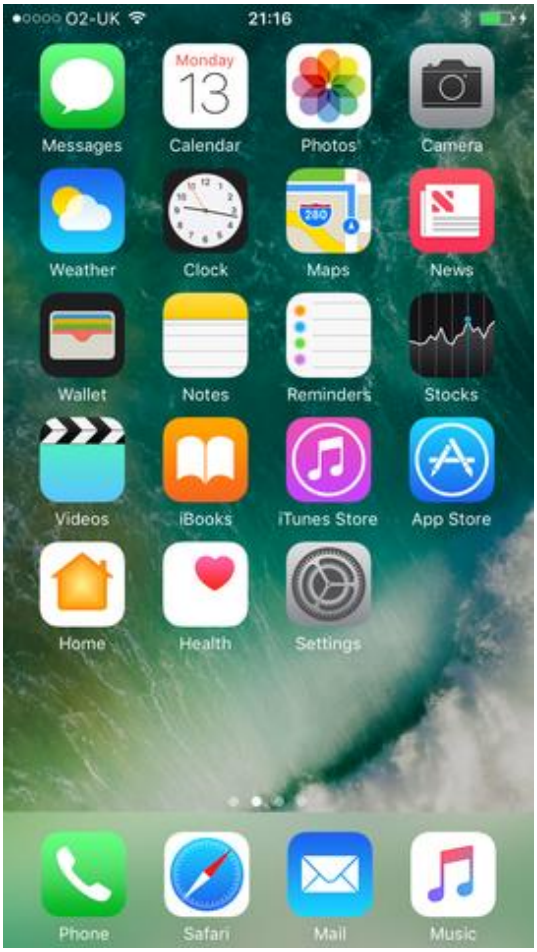

Figura 2 - iOS 10. Pantalla de inicio.

2.1.1.3 Windows Phone

Windows Phone es un sistema operativo para dispositivos móviles desarrollado por *Microsoft* para sustituir a *Windows Mobile* y a *Zune*.

Windows Phone cuenta con una interfaz de usuario derivada del lenguaje de diseño de *Microsoft*, *Modern UI*. Esta interfaz, que nació con el nombre de *Metro*, se compone de un diseño geométrico y colores básicos. Es una interfaz diseñada para funcionar con pantallas táctiles.

A continuación, se realiza un resumen de las versiones que ha tenido *Windows Phone* a lo largo de su existencia [14].

Windows Phone 7

Windows Phone 7 fue la primera versión de este sistema operativo, publicada en 2010.

Esta primera versión incluía una versión móvil de *Internet Explorer 9*, multitarea para aplicaciones propias y para aplicaciones de terceros, integración de *Twitter* y acceso a *Windows Live Skydrive*.

Windows Phone 8

Windows Phone 8 es la segunda generación de este sistema operativo, publicada en 2012.

Windows Phone 8 remplaza la arquitectura basada en *Windows CE* usada en *Windows Phone 7* por *Windows NT kernel*.

Esta versión vino con las siguientes novedades: soporte *NFC*, se incluyó *Skype*, se incluyó *Internet Explorer 10*, se incluyó el modo suspensión en conexiones *Wi-Fi*, modo conducción y optimización de *HTML5*.

Windows Phone 8.1

Windows Phone 8.1 pertenece a la tercera generación de este sistema operativo, publicado en 2014.

Esta versión trajo consigo numerosas novedades y características nuevas como Cortana, un asistente virtual (equivalente a *Siri* y *Google Now*); el centro de notificaciones, sensores para Wi-Fi, datos, batería y almacenamiento; y se introdujo la posibilidad de personalizar la pantalla de inicio.

Asimismo, se añadió un gran número de aplicaciones preinstaladas como *Here Maps*, *Here Transit* y *Here Drive+*, *Xbox Music*, *Xbox Video* y *Xbox Games.*

Windows 10 Mobile

Windows 10 Mobile es el sistema operativo sucesor de *Windows Phone 8.1*. Sin embargo, es distribuido como una versión de *Windows 10*. Fue publicado en 2015.

Windows 10 Mobile continuó con el diseño de interfaz presentado por la línea de *Windows Phone X*, el menú de inicio compuesto por los *Live Tiles* como se puede ver en la Figura 3. A su vez, introdujo mejoras a la hora de personalizar la pantalla de inicio, dando la posibilidad de cambiar el fondo del menú y el listado de aplicaciones y se puede hacer que los *tiles* (azulejos) sean traslucidos.

En esta versión también se ha mejorado la interacción con las notificaciones y a las opciones de activar la conexión *Wi-Fi*, activar el *Bluetooth* o activar el modo avión del centro de notificaciones se han añadido más acciones rápidas.

Asimismo, se ha incorporado el navegador *Microsoft Edge* en Windows 10 Mobile sustituyendo a *Internet Explorer*.

Otras novedades presentes en *Windows 10 Mobile* son: la versión final de *Cortana*, un nuevo teclado y, una novedad muy importante, *Office Mobile* que incluye *Outlook*, *OneDrive*, *OneNote*, *Excel*, *Word* y *PowerPoint*.

Figura 3 - Windows 10 Mobile. Pantalla de inicio.

2.2 Elección del sistema operativo

En este apartado se va a detallar los factores que han llevado a elegir *Android* como plataforma para desarrollar la aplicación.

En primer lugar, se ha tomado en cuenta el número de usuarios y dispositivos presentes en cada uno de los sistemas operativos móviles presentados en el apartado 2.1.

De igual modo, se ha realizado una comparativa referente a los costes de desarrollo de aplicaciones y los costes de publicación de aplicaciones para cada una de las plataformas.

Cuota de mercado

Se ha tenido en cuenta el número de dispositivos vendidos a la hora de elegir el sistema operativo, porque cuanto mayor sea el número de dispositivos que utilizan dicho sistema operativo, mayor será el número de usuario a los que puede llegar la aplicación.

Android continúa siendo la plataforma dominante, liderando la mayoría de mercado con 1240 millones de dispositivos vendidos en 2016. Mientras que *iOS*, sigue siendo la segunda plataforma más extendida, ha vendido 227 millones de dispositivos en 2016. Por último, *Windows Mobile* sigue muy detrás de sus competidores con 11.2 millones de dispositivos vendidos en 2016 [15].

La cuota de mercado que presentan las distintas plataformas puede observarse en la Figura 4. Android ha vendido el 85.2% de los dispositivos vendidos, iOS ha vendido el 13.8% y Windows Mobile ha vendido solamente el 4.2%.

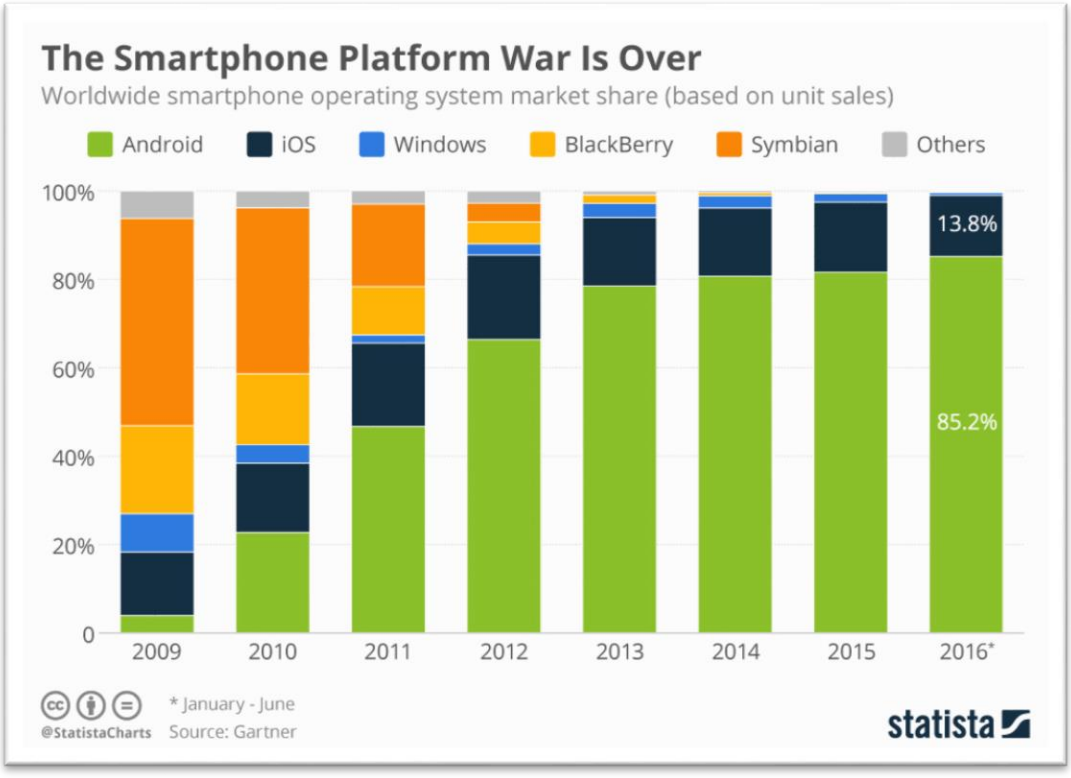

Figura 4 - Cuota de mercado por sistema operativo.

Coste de desarrollo y publicación

Se ha tenido en cuenta los costes que conlleva el desarrollo de una aplicación. Estos costes pueden variar dependiendo de la plataforma en que se desarrolla la aplicación. Por ello, se ha realizado una comparativa del coste de desarrollo de una aplicación para *Android*, *iOS* y *Windows*.

Asimismo, se ha tenido en cuenta los costes que conlleva la publicación de una aplicación. Estos costes pueden variar dependiendo de la plataforma en que se desarrolla la aplicación. Por ello, se ha realizado una comparativa del coste de publicación de una aplicación en *Google Play*, *App Store* y en la tienda de aplicaciones de *Windows*.

Las aplicaciones para *Android* se desarrollan con el SDK (*Software Development Kit*) de *Android* que es gratuito y está disponible para *Windows*, *Mac* y *Linux*. *Google* realiza un único cobro de 25$ para obtener una cuenta de desarrollador en *Google Play*, que permite publicar aplicaciones. Las aplicaciones gratuitas se distribuyen sin costes extra. Mientras que *Google* se queda el 30% de los ingresos de las aplicaciones de pago [16].

Apple realiza cobros de 99$ al año para unirse al programa de desarrolladores de *iOS* (*iOS Developer Program*) que permite publicar aplicaciones para iPhone, iPad y iPod. Las aplicaciones gratuitas no tienen costes adicionales. Mientras que *Apple* se queda el 30% de los ingresos de las aplicaciones de pago. Para desarrollar aplicaciones se necesita un equipo con *Mac Os X*, el software de desarrollo viene incluido en el precio del programa de desarrolladores de *iOS* [17].

Microsoft realiza un único cobro de 19$ para una cuenta individual y 99$ para una cuenta de empresa para unirse al centro de desarrolladores de *Windows* (*Windows Dev Center*) que permite publicar aplicaciones para Windows 10 Mobile y Xbox. Las aplicaciones gratuitas no tienen costes adicionales. Mientras que *Microsoft* se queda el 30% de los ingresos de las aplicaciones de pago. Para desarrollar aplicaciones se necesita utilizar *Visual Studio* en un equipo con *Windows 7*, *Windows 8* o *Windows 10* [18].

En la Tabla 1 se muestran los detalles relativos a los costes de desarrollo y publicación de aplicaciones en *Android*, *iOS* y *Windows*.

Plataforma	Coste de desarrollo	Coste de publicación
Android	Gratis	25$, pago único
iOs	99$ anuales	Incluidos en el coste de desarrollo
Windows	Gratis	Individual: 19$, pago único Empresa: 99$

Tabla 1. Costes de desarrollo y publicación de aplicaciones.

Conclusión de los resultados

A partir de los datos presentados acerca del número de dispositivos por plataforma, se observa que *Android* es el sistema operativo móvil más utilizado, con mucha diferencia. El segundo sistema operativo más utilizado es *iOS*, con buena cuota de mercado, pero bastante detrás de *Android*.

Según los datos presentados en la Tabla 1 relativos a los costes de desarrollo y publicación de aplicaciones por plataforma, se observa que:

- Los costes de desarrollo en *Android* y *Windows* son gratuitos. Mientras que, en iOs son 99$ anuales.
- Los costes de publicación en iOs vienen incluidos en los 99$ anuales. Android y Windows presentan pagos únicos de 25$ y 19$, respectivamente.

Se ha determinado que la mejor opción para desarrollar la aplicación es Android, tomando en cuenta la cuota de mercado que tiene en este momento (2016), muy superior a sus competidores, y el coste de desarrollar y publicar una aplicación, es decir, un único pago de 25 dólares.

2.3 Sistemas de diálogo hablado

Los sistemas de diálogo hablado (*Spoken Dialogue Systems*) permiten la comunicación de una persona con un computador mediante lenguaje natural expresado de manera oral [19].

Un sistema de diálogo ideal debe reconocer el habla espontánea sin dificultades, comprender todo tipo de enunciados sin restricciones de contenido, proporcionar respuestas con sentido, gramaticalmente bien formadas y pragmáticamente adecuadas; y permitir una interacción multimodal a través del habla y de la imagen [20].

Desarrollar un sistema de diálogo ideal aún no es posible, debido a la necesidad de numerosas fuentes de conocimiento y las limitaciones de las tecnologías actuales. Sin embargo, la continua progresión en la tecnología del habla ha permitido que sean posibles los sistemas de comunicación persona-máquina mediante la voz [21].

2.3.1 Arquitectura de un sistema de diálogo

Un sistema de diálogo hablado se puede desarrollar mediante una arquitectura modular como la presentada en la Figura 5 [21].

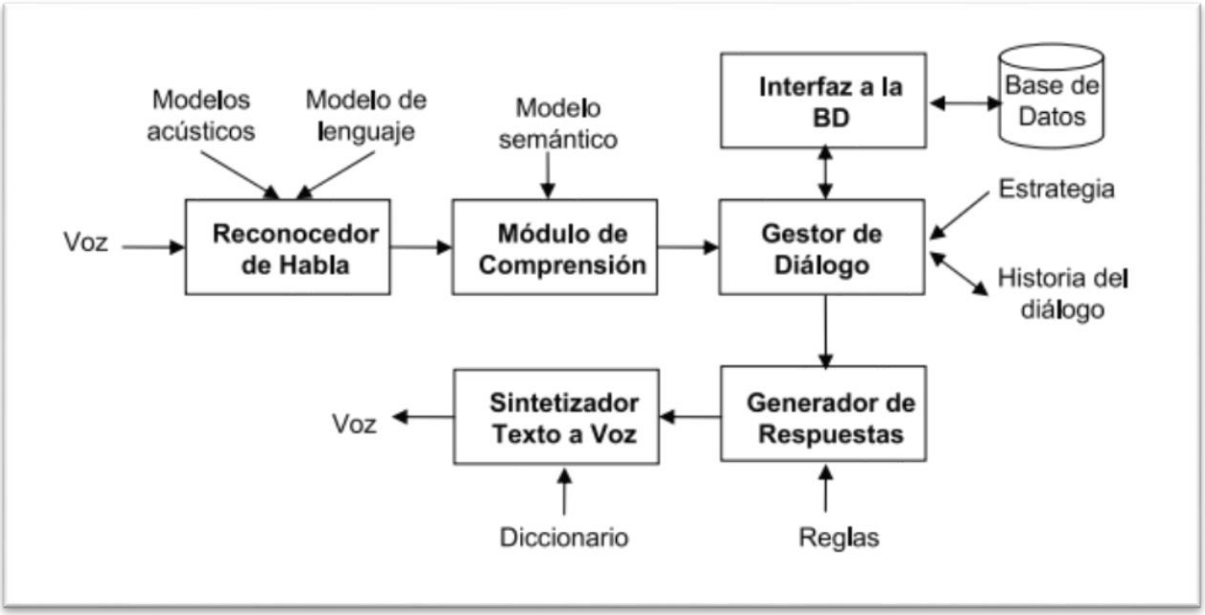

Figura 5 - Arquitectura modular de un sistema de diálogo.

Se va a detallar los módulos que componen la arquitectura presentada en la Figura 5 en los siguientes apartados [21].

Módulo de reconocimiento automático del habla

El módulo de reconocimiento automático del habla se encarga de captar y reconocer la señal de voz pronunciada por el usuario. El módulo procesa la señal de voz y produce la secuencia de palabras reconocidas.

Módulo de comprensión del habla

Los datos de entrada para el módulo de compresión del habla son las secuencias de palabras generadas en el módulo de reconocimiento automático del habla.

A partir de estos datos el sistema obtiene una representación semántica de su significado.

Gestor de diálogo

El gestor de diálogo es el módulo más importante de un sistema de diálogo hablado, interacciona con los módulos de entrada del sistema y con los módulos que generan la salida.

El principal objetivo del gestor de diálogo es gestionar el flujo de datos que provienen de los módulos anteriores. Asimismo, se encarga de decidir qué acciones se van a realizar en cada interacción con el usuario, de solicitar información a los módulos que controlan el acceso a la base de datos y de generar la respuesta del sistema apropiada al estado actual del diálogo.

Módulo de consulta a la base de datos de la aplicación

El módulo de consulta a la base de datos de la aplicación recibe peticiones de consulta a la base de datos que envía el gestor de diálogo, proceso las peticiones y devuelve el resultado al gestor de diálogo.

Módulo de generación de respuestas

El módulo de generación de respuestas recibe la respuesta proporcionada por el gestor de diálogo.

A partir de esta respuesta el módulo de generación de respuestas genera una frase gramaticalmente correcta, lo mas cerca posible al lenguaje natural.

Sintetizador de voz a texto

El sintetizador de voz a texto recibe la respuesta del sistema como texto en lenguaje natural y genera una señal de audio que se envía al usuario.

2.3.2 Sistemas de diálogo multimodal

Un sistema de diálogo multimodal (multimodal dialogue system) tiene como objetivo principal superar las barreras de la interacción basada exclusivamente en el habla [19].

En una interacción multimodal el usuario tiene alternativas a la comunicación mediante voz para comunicarse con un computador o máquina. Estas alternativas pueden ser diversos dispositivos de entrada como teclado, ratón, pantalla táctil, cámara, etc.

De la misma manera, un sistema multimodal puede hacer uso de diferentes canales de salida para proporcionar información a los usuarios como voz, texto, imágenes, etc.

2.4 Análisis de aplicaciones similares

En este apartado se realiza un estudio de aplicaciones que pertenencen al campo del *m-learning* y del *e-learning* que guardan similitud con la aplicación, que se va a llevar a cabo en este proyecto, en cuanto a características, modo de uso, temática, finalidad, etc. Se realiza este análisis con el objetivo de determinar sus principales características, extraer ideas para requisitos y diseño, y de esta manera dar los primeros pasos en el desarrollo del proyecto.

Quizlet

Quizlet es una aplicación y un sitio web diseñado para que los usuarios puedan estudiar idiomas, vocabulario, historia o ciencia. El aprendizaje se realiza mediante fichas educativas (ver Figura 6) hechas por el propio usuario para un tema concreto o puede utilizar las fichas creadas por otros usuarios. Después de haber aprendido o memorizado las tarjetas el usuario procede a realizar pruebas o exámenes para confirmar los nuevos conocimientos adquiridos [22].

Los modos de aprendizaje son el modo *aprender* que pone a prueba la memoria del usuario y el modo *combinar* que es una prueba contrarreloj. Dependiendo del tipo de prueba la respuesta se da escribiendo con el teclado, eligiendo una respuesta de entre un conjunto de posibilidades o eligiendo entre verdadero y falso. Al finalizar cada una de las pruebas se presentan los resultados obtenidos.

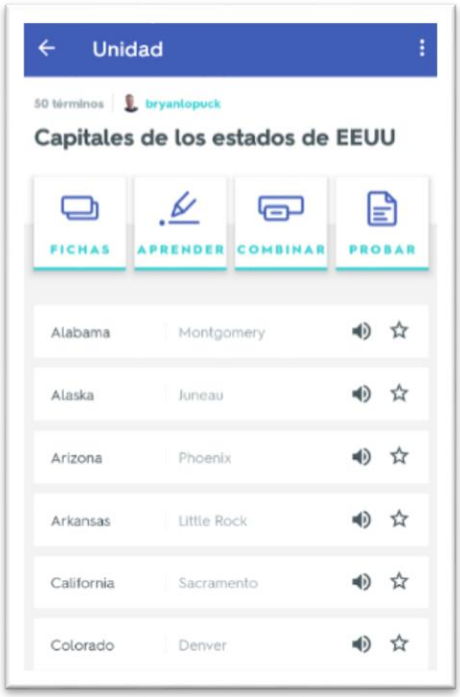

Figura 6 - Quizlet.

Super Flashcards

Esta aplicación, al igual que *Quizlet*, se basa en aprendizaje mediante fichas educativas. *Super Flashcards* esta pensada para que los usuarios puedan profundizar sus conocimientos en diversas áreas como ciencias, idiomas, historia, etc. La aplicación permite importar conjuntos de tarjetas hechas en *Quizlet*, de esta manera ofrece un contenido bastante extenso.

En primer lugar, el usuario selecciona la materia o el tema que desea aprender, entonces la aplicación le mostrará una serie de tarjetas que ayudarán al usuario a memorizar conceptos clave de dicha materia. A continuación, el usuario puede poner a prueba sus nuevos conocimientos mediante preguntas y tests, relacionados con el contenido de las tarjetas.

Las preguntas que ofrece *Super Flashcards* son de tipo test, preguntas de verdadero o falso y preguntas de respuesta mediante teclado. Los modos de aprendizaje son: modo *normal* en el que se muestran las preguntas en orden aleatorio y el modo *endless* en el que muestran únicamente las preguntas que el usuario ha contestado de manera incorrecta.

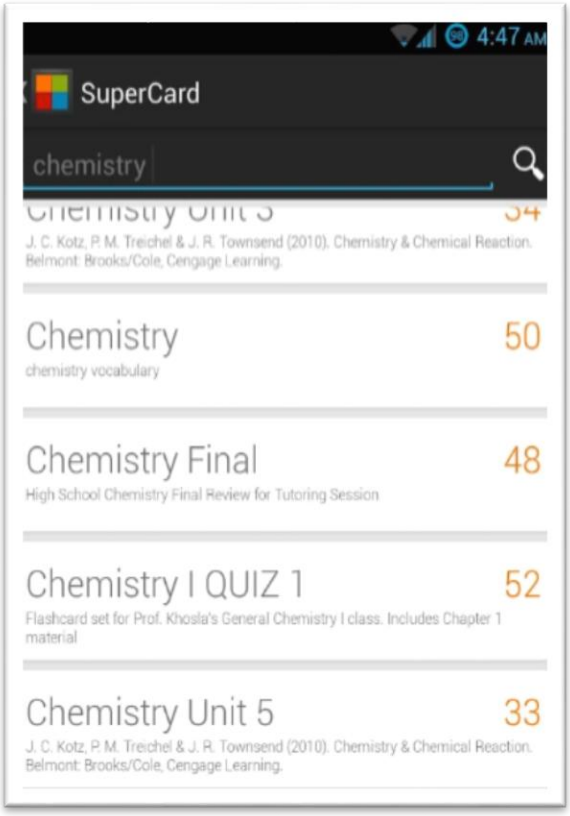

Figura 7 - Super Flashcards.

Duolingo

Duolingo es una plataforma de aprendizaje de idiomas que se compone de un sitio web y una aplicación para móviles. La aplicación ofrece 66 cursos de idiomas disponibles en 23 idiomas [23].

Duolingo proporciona lecciones de escritura, lecciones de dictado y prácticas orales para usuario avanzados. La idea principal de la aplicación es adquirir habilidades en el idioma seleccionado, para ello se deben completar lecciones. Para completar una lección es necesario obtener diez puntos, acertar una pregunta suma un punto y fallar una pregunta resta un punto. Cada vez que se completa una lección el usuario recibe puntos de experiencia para poder acceder a niveles más avanzados.

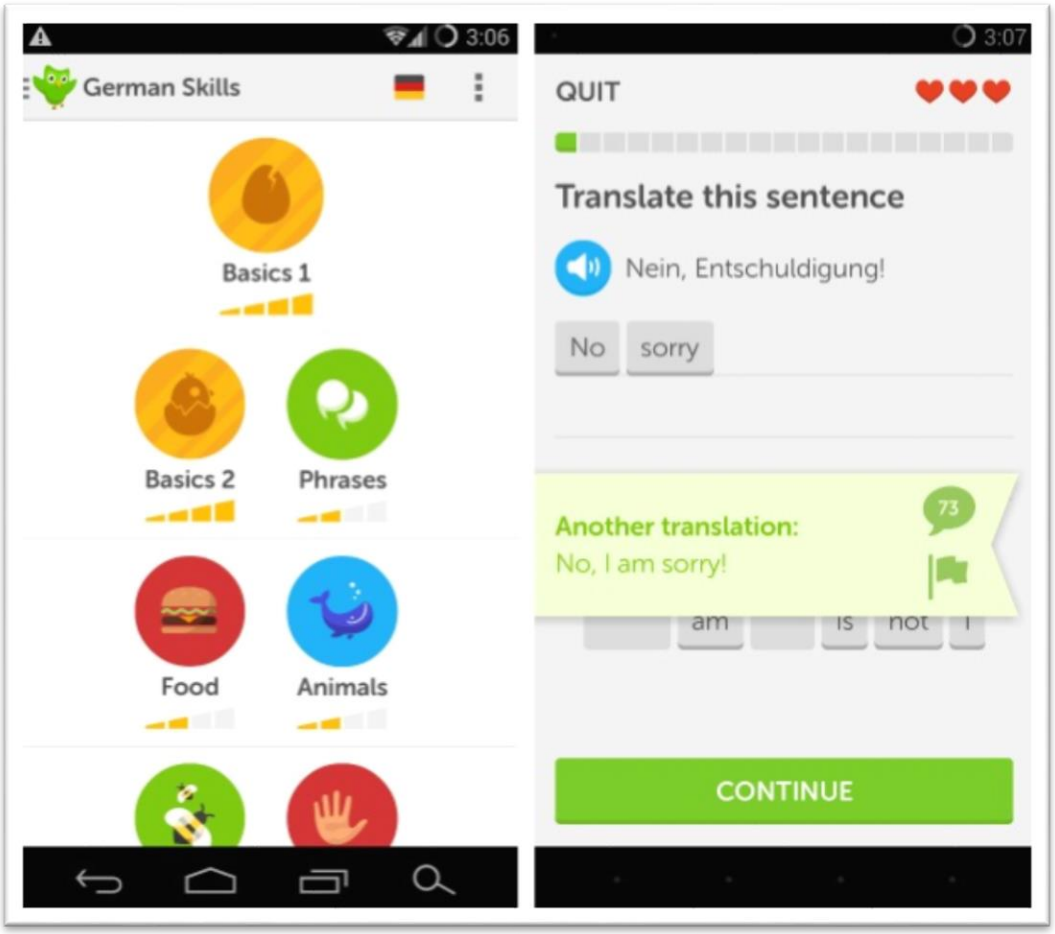

Figura 8 - Duolingo.

Preguntados

Preguntados es un juego para dispositivos móviles que está inspirado en el juego de mesa *Trivial Pursuit*. Esta aplicación enfrenta a dos usuarios, vía internet, que deben demostrar sus conocimientos sobre ciencia, arte, historia, deporte, geografía y entretenimiento. El objetivo es lograr responder correctamente una pregunta de cada uno de los campos mencionados antes que el oponente.

Las preguntas que se realizan en el juego son de tipo test o de opción múltiple, es decir, se muestra la pregunta y cuatro posibles respuestas. El usuario elige una de ellas y la aplicación comprueba si es correcta o no. Las partidas se realizan por turnos, cuando un usuario falla el turno pasa a su contrincante. Sin embargo, si el usuario acierta, continúa contestando preguntas hasta que falle o gane.

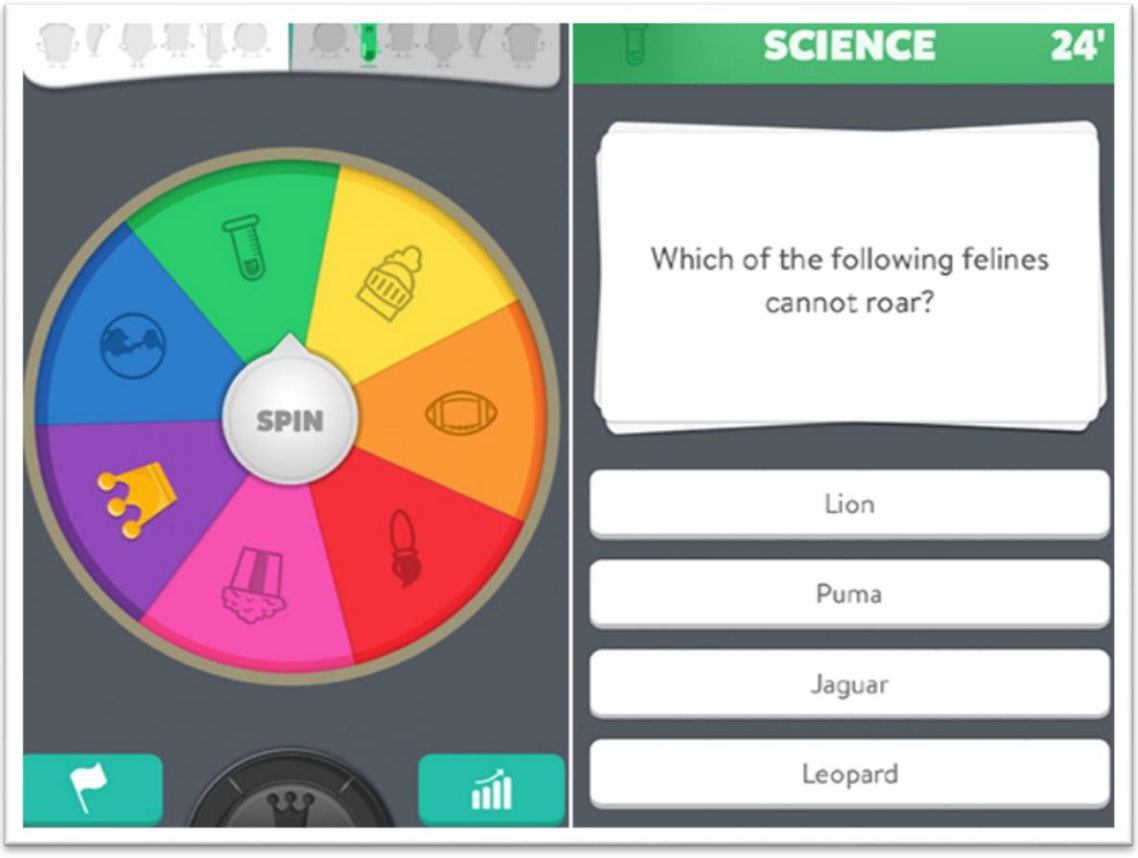

Figura 9 - Preguntados.

Capítulo 3

3. Descripción del sistema

En este tercer capítulo se realiza un análisis general del sistema, así como de las distintas tecnologías y herramientas software utilizadas para desarrollar el proyecto. Asimismo, se especifican los requisitos del sistema.

3.1 Descripción general

El sistema que compone el contenido principal de este documento está formado, a su vez, por diferentes subsistemas que interactúan y cooperan entre ellos. La Figura 10 muestra la arquitectura cliente-servidor del sistema y los distintos subsistemas que componen el lado cliente y el servidor, así como, la comunicación que existe entre ellos.

- El lado del cliente está compuesto por la aplicación Android.
- El lado del servidor está compuesto por el servidor web y la base de datos.

La aplicación Android (cliente) solicita realizar una acción sobre la base de datos, para ello envía una petición HTTP al servidor web.

Los ficheros PHP forman el servicio web que, a su vez, hace de punto intermedio (*middleware*) entre la aplicación y la base de datos. El servicio web convierte la petición HTTP en una sentencia SQL que se envía a la base de datos.

La base de datos procesa la sentencia SQL y crea, recupera, almacena o borra la información solicitada. La base de datos devuelve los resultados al servidor PHP.

El servidor web recibe la información devuelta por la base de datos y convierte estos datos en formato JSON y se los envía a la aplicación cliente.

La aplicación obtiene la información en formato JSON, la analiza y realiza las acciones que correspondan.

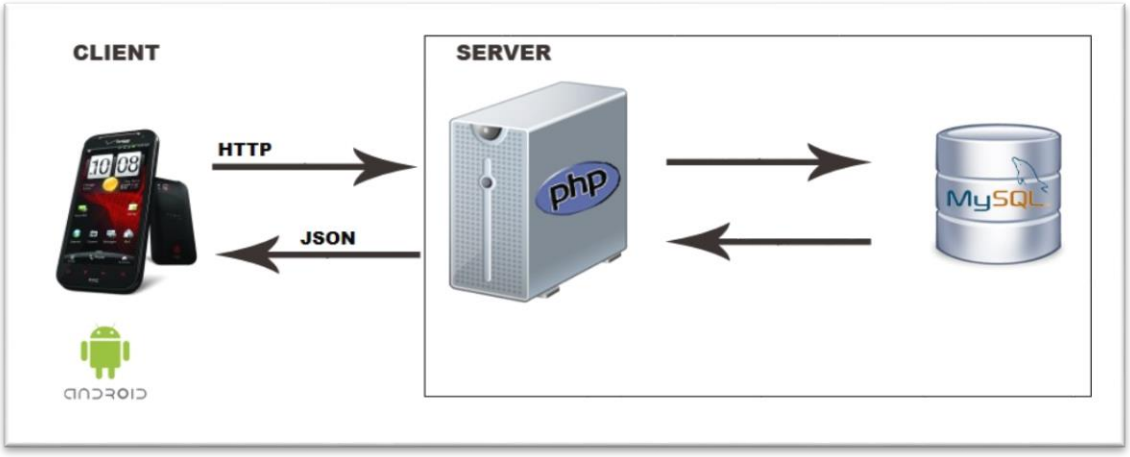

Figura 10 - Arquitectura de la aplicación.

En los siguientes puntos se habla en más detalle de cada uno de los componentes, tecnologías y herramientas del sistema que se está desarrollando.

3.2 Plataforma Android

Android es un software de código abierto creado para una amplia gama de dispositivos. Los objetivos principales de *Android* son la creación de una plataforma de software abierto disponible para que los desarrolladores puedan llevar a cabo sus ideas innovadoras y para mejorar la experiencia móvil de los usuarios [24].

A lo largo de este punto se va a explicar la arquitectura de la plataforma *Android*, se van a detallar las componentes que constituyen una aplicación de *Android*, se va a definir el entorno de desarrollo de aplicaciones y se va a describir el proceso de desarrollar una aplicación para dicha plataforma.

3.2.1 Arquitectura de Android

La arquitectura de *Android* está formada por cinco capas que facilitan la tarea del desarrollador a la hora de crear aplicaciones.

La Figura 11 [24] muestra de manera visual las capas que conforman la arquitectura de Android. Android presenta una arquitectura en forma de pila software. Cada capa de la pila, y los elementos correspondientes a cada capa, están altamente integrados para proporcionar el desarrollo óptimo de las aplicaciones y el entorno de ejecución óptimo para dispositivos móviles [25].

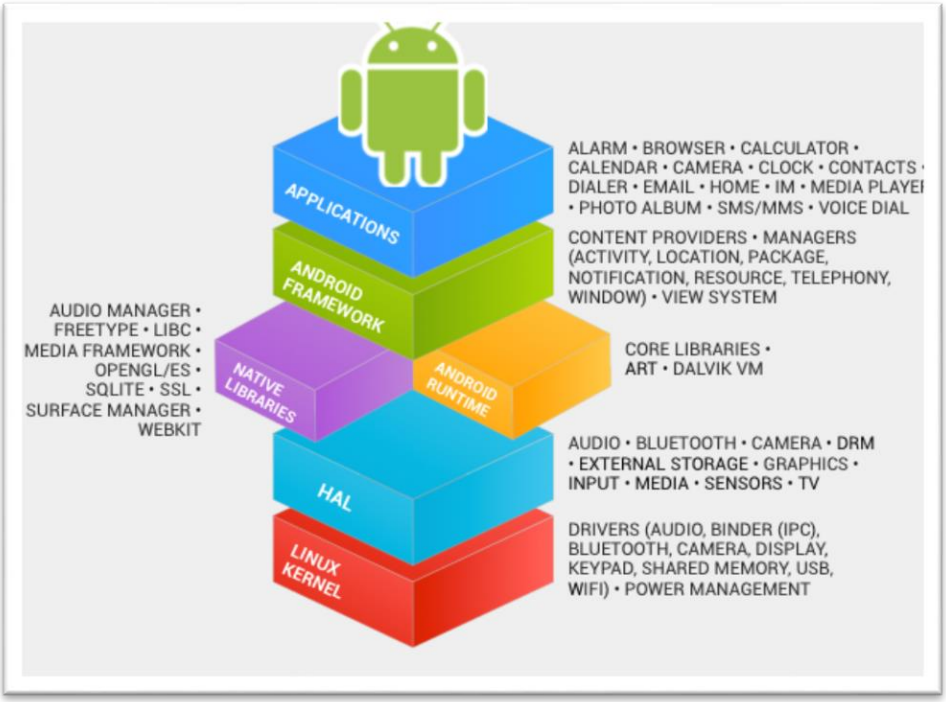

Figura 11 - Arquitectura de Android.

En las siguientes subsecciones, se explican las capas que componen la arquitectura de Android [26].

Kernel de Linux

La capa de más bajo nivel corresponde al kernel de Linux. El núcleo de Android está basado en basado en núcleo de Linux 2.6.

El kernel de Linux proporciona servicios básicos como gestión de procesos, gestión de memoria y gestión de dispositivos a najo nivel.

Hardware Abstraction Layer

La capa de abstracción de hardware (*Hardware Abstraction Layer*) proporciona interfaces estándar que exponen las capacidades del hardware del dispositivo al *Java API Framework* de nivel superior.

La capa de abstracción de hardware se compone de múltiples módulos de librerías, donde cada una implementa una interfaz para un componente hardware específico, como la cámara o el módulo de *bluetooth*. Cuando se realiza una llamada para

acceder a un dispositivo hardware, el sistema carga la librería correspondiente para ese componente.

Android Runtime

En dispositivos que utilizan la versión 5.0 de *Android* o posterior, cada aplicación se ejecuta en su propio proceso con su propia instancia de *Android Runtime* (entorno de ejecución de *Android*). En versiones anteriores a Android 5.0, el entorno de ejecución utilizado era *Dalvik*.

Android Runtime está optimizado para ejecutar múltiples máquinas virtuales en dispositivos de memoria baja, para ello utiliza los ficheros DEX que es un formato *bytecode* diseñado especialmente para *Android*.

Librerías nativas

Android incluye en conjunto de librerías escritas en C y C++ que se utilizan para muchos de los componentes principales del sistema *Android*.

En la Tabla 2 se muestran las librerías nativas más importantes [27].

Librería	Función
Freetype	Permite la representación de fuentes mediante mapa de bits y vectores.
Libc	Librería del sistema estándar de C optimizada para Android.
Media Framework	Da soporte a la reproducción y grabación de audio y vídeo.
OpenGL/ES	Gestiona la renderización de gráficos en 2D y 3D para múltiples aplicaciones.
SQLite	Base de datos relacional liviana disponible para el uso de cualquier aplicación.
SSL	Establece comunicaciones seguras mediante el uso del protocolo SSL.
Surface Manager	Se encarga de la composición de los elementos de interfaz y de navegación de la pantalla.
Webkit	Proporciona un motor web para el navegador.

Tabla 2. Librerías nativas de Android.

Android Framework

El conjunto completo de servicios del sistema operativo *Android* está disponible mediante APIs escritas en el lenguaje Java. Estas APIs simplifican el desarrollo de aplicaciones mediante la reutilización de sistemas y servicios del sistema.

En la tabla 3 se explican las APIs más importantes de la capa *Android Framework* [26].

APIs	Función
Activity Manager	Gestiona el ciclo de actividad de las aplicaciones y proporciona la pila de navegación.
Content Providers	Permite a las aplicaciones acceder a los datos de otras aplicaciones, como la aplicación de contactos o compartir sus propios datos.
Location Manager	Permite conocer la ubicación geográfica del dispositivo mediante GPS o internet.
Notification Manager	Permite a todas las aplicaciones publicar notificaciones en la barra de estado.
Resource Manager	Gestiona el acceso a los elementos de una aplicación que complementan el código, como cadenas de texto, sonidos, imágenes, etc.
Sensor Manager	Gestiona el acceso a los sensores hardware presentes en el dispositivo, los cuales pueden ser: acelerómetro, sensor de luminosidad, brújula, giroscopio, etc.
View System	Proporciona los elementos para construir interfaces de usuario como listas, botones, cuadros de texto, etc.

Tabla 3. APIs del Framework de Android.

Aplicaciones

La capa de aplicaciones es la capa de más alto nivel. Esta capa contiene todas las aplicaciones instaladas en el dispositivo, tanto aplicaciones del sistema como aplicaciones de terceros.

Android tiene un conjunto de aplicaciones nativas para el correo electrónico, mensajes, calendario, navegador de internet, contactos, etc. Sin embargo, estas pueden remplazarse por aplicaciones de terceros.

3.2.2 Componentes de una aplicación

Las aplicaciones se componen de cuatro elementos. Cada uno de estos componentes tiene una finalidad diferente y ciclo de vida distinto que indica cómo se crea y destruye el componente.

Una aplicación en Android tiene que declarar sus *activities*, puntos de entrada, puntos de salida, capas, permisos e *intents* en el *AndroidManifest*. En las siguientes subsecciones se va a realizar un resumen de los cuatro elementos que componen una aplicación *Android* [28].

3.2.2.1 Activity

Un componente *activity* representa a una única pantalla con su interfaz de usuario asociada. Es el componente más habitual en una aplicación Android.

Este componente se implementa mediante la clase homónima, *Activity*. Las diferentes pantallas que componen una aplicación están representadas mediante un *activity*, de esta manera al navegar entre pantallas, en realidad, se duerme un *activity* y se lanza otra. En la Figura 12 [29] se puede observar el ciclo de vida de un *activity*.

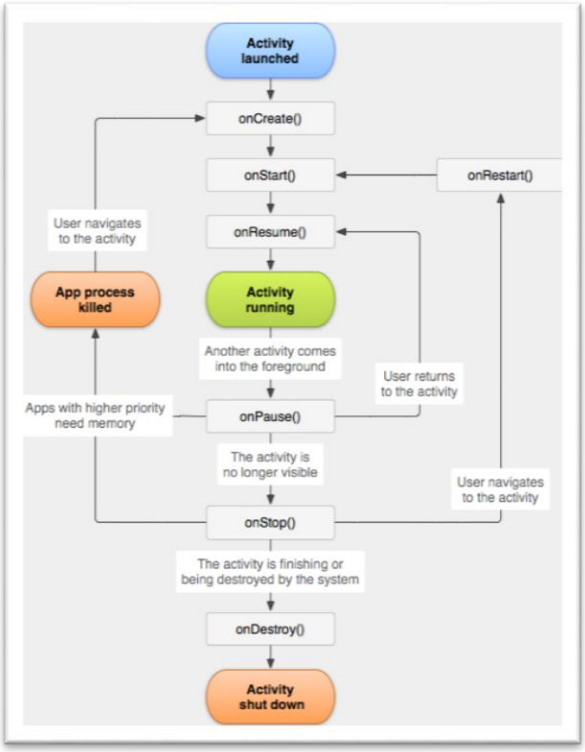

Figura 12 - Ciclo de vida de un componente Activity.

3.2.2.2 Services

Un *service* es un componente que se ejecuta en segundo plano para realizar tareas de larga duración o para realizar trabajos para procesos remotos. Un *service* no posee una interfaz de usuario asociada. Un service se implementa como subclase de la clase *Service*.

Por ejemplo, un *service* puede reproducir música mientras el usuario utiliza otra aplicación.

3.2.2.3 Content providers

Un *content provider* permite a las aplicaciones almacenar datos en un fichero del sistema, una base de datos SQLite o cualquier otro almacenamiento persistente. A través del *content provider*, otras aplicaciones pueden consultar o modificar los datos almacenados.

Por ejemplo, Android proporciona un content provider que gestiona la información de los contactos del usuario. De esta manera, cualquier aplicación con los permisos correspondientes puede consultar la información sobre una persona.

Un *content provider* es también útil para leer y escribir datos privados de una aplicación. Un *content provider* se implementa como subclase de la clase *ContentProvider*.

3.2.2.4 Broadcast receivers

Un *broadcast receiver* es un componente que responde a mensajes de difusión en todo el sistema. Aunque un *broadcast receiver* no tiene interfaz de usuario asociada, es necesario que cree una notificación para alertar al usuario que ha ocurrido un evento.

El sistema envía muchos mensajes de difusión, por ejemplo, un anuncio de difusión sobre pantalla apagada, batería baja o captura de pantalla. Las aplicaciones también pueden enviar mensajes de difusión.

Un *broadcast receiver* se implementa como subclase de la clase *BroadcastReceiver*.

3.2.3 Entorno de desarrollo

A lo largo de este punto se va a detallar el proceso que se debe seguir para preparar el entorno de desarrollo de aplicaciones Android en Mac OS X. Para desarrollar aplicaciones se tiene que tener instalado el siguiente software [30] [31].

1. Java Development Kit (JDK)
2. Android Studio IDE
3. Software Development Kit (SDK)

3.2.3.1 Java Development Kit (JDK)

Antes de instalar las herramientas de desarrollo para Android, es necesario instalar Java Development Kit (JDK).

Java no viene instalado en las versiones más recientes de Mac Os X. Para asegurarse si Java está instalado se tiene que abrir el *Terminal* e introducir el comando "java -version".

- En el caso de tener una versión de Java instalada en la máquina, el Terminal mostrará el mensaje contenido en la Figura 13 [30].

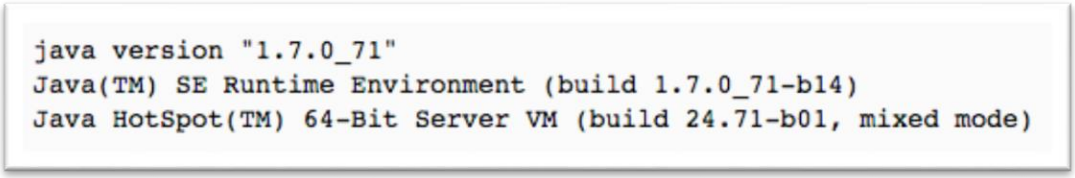

Figura 13 - Terminal. Java instalado.

- En el evento de no tener una versión de Java instalada, el Terminal muestra el mensaje contenido en la Figura 14 [30]. Entonces, es necesario descargar la última versión estable del JDK, en este caso, JDK 8. En la web de descarga del JDK (http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html) se debe seleccionar el sistema operativo y la versión deseados.

  Una vez descargado el instalador (archivo dmg) se ejecuta y aparece la ventana del asistente de instalación (ver Figura 15 [31]). Por último, se deben seguir los pasos marcados por el asistente de instalación.

Figura 14 - Terminal. Java no instalado.

Figura 15 - Asistente de instalación de JDK.

3.2.3.2 Android Studio IDE

Android Studio vs Eclipse

Se ha elegido Android Studio sobre Eclipse porque Android Studio es el IDE (*Integrated Development Environment*) oficial de Android.

De hecho, Google ha dejado de dar soporte oficial al plugin ADT (*Android Developer Tools*) en Eclipse a finales del año 2015 [32].

Asimismo, se ha elegido Android Studio por ser un software creado específicamente para desarrollar aplicaciones Android.

Instalación de Android Studio

Android Studio se descarga desde la web oficial de desarrolladores de Android https://developer.android.com/studio/index.html.

Una vez descargado el instalador de Android Studio, se ejecuta este archivo de formato dmg y aparece el asistente de instalación de Android Studio. A continuación, se debe arrastrar y soltar la aplicación a la carpeta *Applications* para que comience la instalación (ver Figura 16 [31]).

Puede ser necesario dar permiso al sistema para que instale aplicaciones de terceros. Para ello, simplemente se tiene que acceder al menú de seguridad y privacidad siguiente este recorrido, *Preferencias del sistema > Seguridad y privacidad > General* y seleccionar la opción *Permitir aplicaciones de Mac App Store y desarrolladores identificados*.

Figura 16 - Instalación de Android Studio.

3.2.3.3 Software development kit de Android

El SDK de Android viene incluido en Android Studio, que se instaló en el apartado 3.2.3.2, por lo tanto, no es necesario descargar el SDK. Sin embargo, en el caso de Eclipse con ADT era necesario descargar el SDK aparte.

El SDK incluye los paquetes básicos, pero no incluye todo lo necesario para comenzar a desarrollar aplicaciones. Por lo tanto, es necesario añadir los paquetes que faltan. Para ello, se accede al *SDK Manager* mediante la ruta *Tools > Configure > SDK Manager* desde la ventana principal de Android Studio.

Los paquetes imprescindibles para desarrollar aplicaciones y que deben descargarse sin falta son (ver Figura 17 [31]):

- Android SDK Tools.
- Android SDK Platform-tools.
- Android SDK Build-tools (versión más reciente).
- SDK Platform.
- Android Support Repository.
- Android Support Library.

Como último paso se debe aceptar los términos de licencia e instalar cada uno de los paquetes.

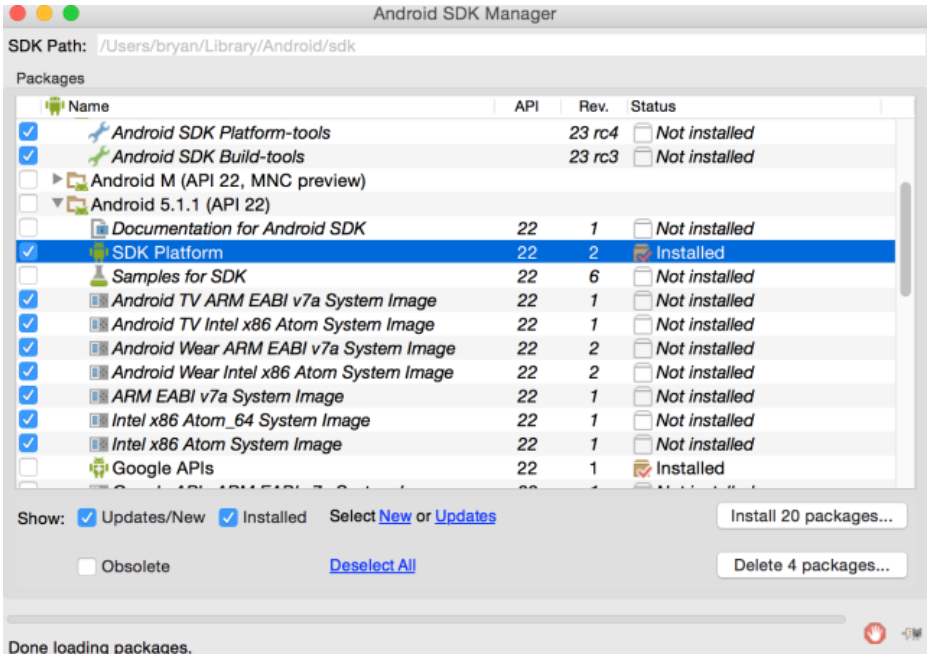

Figura 17 - Android SDK Manager.

3.2.4 Desarrollo de una aplicación

3.2.4.1 Crear un proyecto en Android Studio

Para crear un proyecto en Android Studio se tienen que seguir los siguientes pasos:

1. Iniciar Android Studio, se carga su pantalla de bienvenida. En esta pantalla se tiene que seleccionar "*Start a New Android Studio Project*".
2. Configurar el nombre de la aplicación, el nombre del proyecto y la ubicación del proyecto.
    a. Nombre de la aplicación: El nombre de la aplicación elegido es *TestMe*.
    b. Nombre del proyecto: Es el nombre que se le da al proyecto, en este caso, utilizamos el mismo que el nombre de la aplicación.
3. Seleccionar el tipo de dispositivo para el cual se va a desarrollar la aplicación y SDK mínimo.
    a. Dispositivo objetivo: La aplicación que se va a desarrollar está pensada para smartphones, por lo tanto, se elige la opción Phone and Tablet.
    b. SDK mínimo: Se elige el API 19 (Android 4.4 KitKat) que permite llegar a un 73.9% de los dispositivos y además nos permite implementar todas las funcionalidades necesarias en la aplicación. (ver Figura 18).
4. Añadir un *activity* al proyecto para comenzar a desarrollar.
    a. Se añade un "Empty Activity" para comenzar el desarrollo sobre una aplicación en blanco.
5. Configurar el nombre del *activity* y el nombre de su *layout* asociado.
    a. Nombre del *activity*: nombre que se da a la primera pantalla que se va a implementar, en este caso *MainActivity*.
    b. Nombre del layout: nombre que se da al archivo de interfaz que está asociado al *activity*, en este caso, *activity_main*.
6. Comenzar a desarrollar la aplicación (ver Figura 19).

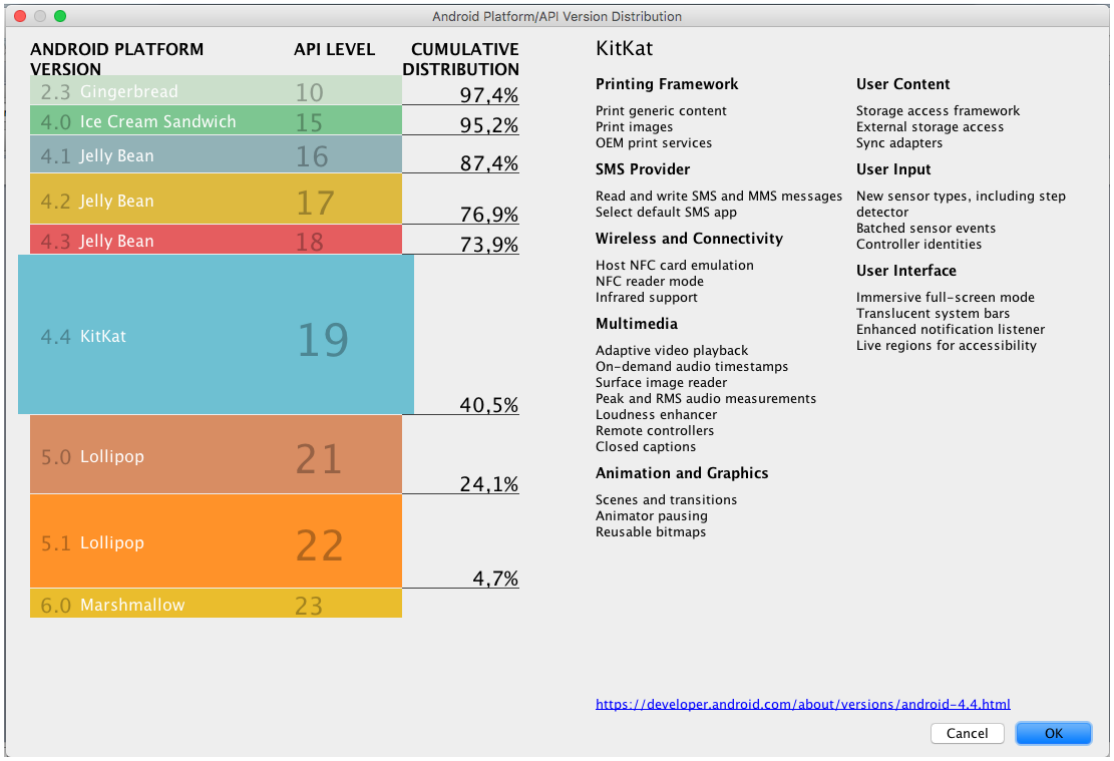

Figura 18 - SDK mínimo.

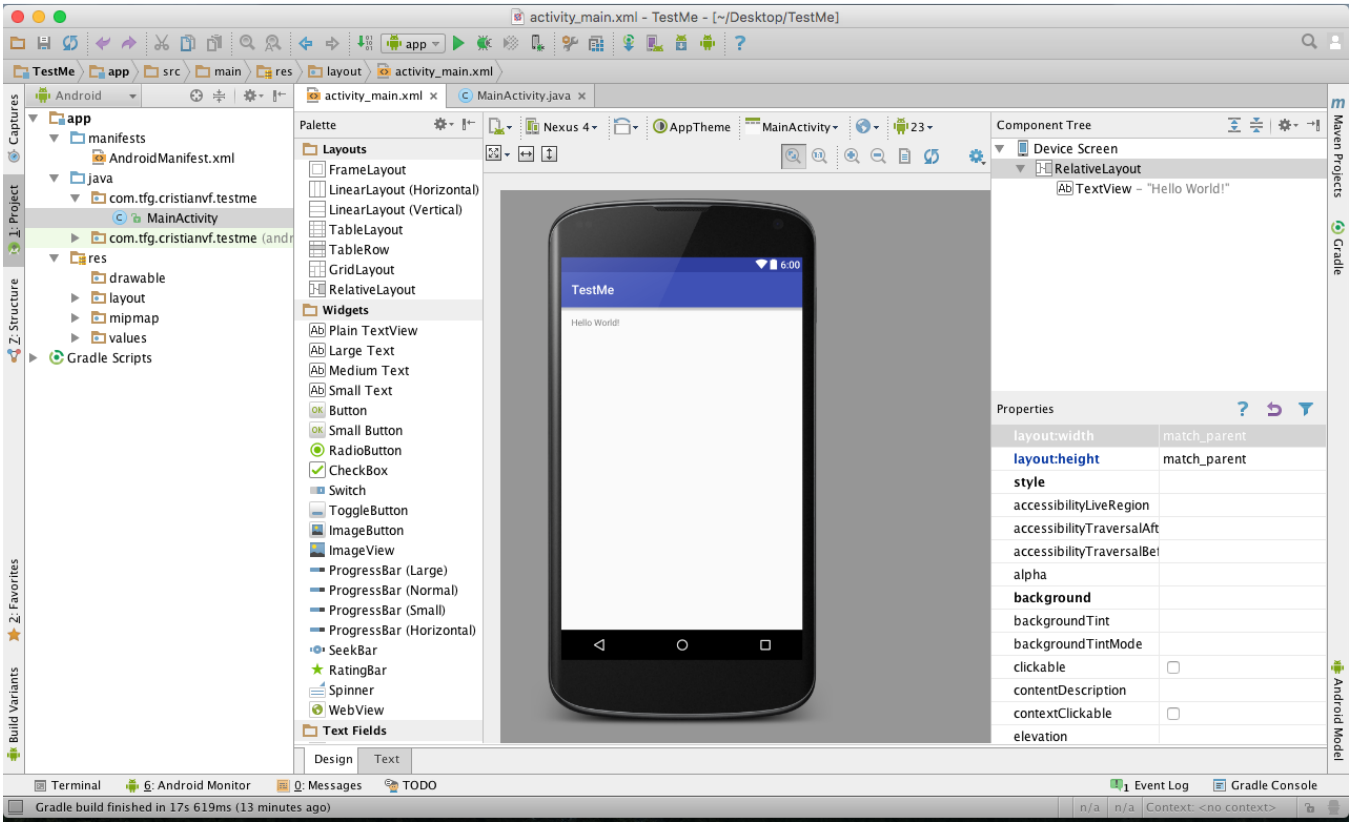

Figura 19 - Proyecto nuevo en Android Studio.

3.2.4.2 Estructura de un proyecto en Android Studio

La estructura de un proyecto en Android Studio se puede presentar desde dos perspectivas:

- Perspectiva del proyecto: Se muestra la estructura del proyecto reflejando la jerarquía de ficheros tal como se encuentran en disco.
- Perspectiva de Android: Se muestra la estructura del proyecto organizada en módulos y por el tipo de archivos para simplificar la navegación en el proyecto.

Se ha elegido la perspectiva de Android porque es más intuitiva para el desarrollador, facilita el trabajo con ficheros del mismo tipo y esconde directorios y ficheros que no suelen ser utilizados para el desarrollo de aplicaciones (ver Figura 20). La estructura que presenta la perspectiva de Android es la siguiente:

- manifests: contiene el fichero *AndroidManifest.xml*.
- java: Contiene los ficheros de código java que componen la aplicación, separados por paquetes.
- res: Contiene todos los recursos, que no son código, utilizados por la aplicación.
    - drawable: Este subdirectorio contiene los archivos multimedia que utilizará la aplicación como timbres, sonidos, canciones, imágenes animadas, etc.
    - layout: Contiene los ficheros *xml* que definen la interfaz de usuario. Cada uno de estos ficheros está asociado a una fichero *java* (*activity*).
    - mipmap: Este subdirectorio contiene los iconos utilizados en la aplicación como el icono de bienvenida, iconos del *action bar* o iconos añadidos a botones para hacerlos más intuitivos. Asimismo contiene las imágenes que puede ser utilizadas como fondo de pantalla o como desee el desarrollador.
    - values: Contiene ficheros *xml* en los que se definen diferentes atributos como colores (*colors.xml*), estilos (*styles.xml*), medidas (*dimens.xml*) y cadenas de texto predefinidas (*strings.xml*). Todos estos atributos están destinados a utilizarse en la interfaz de usuario.

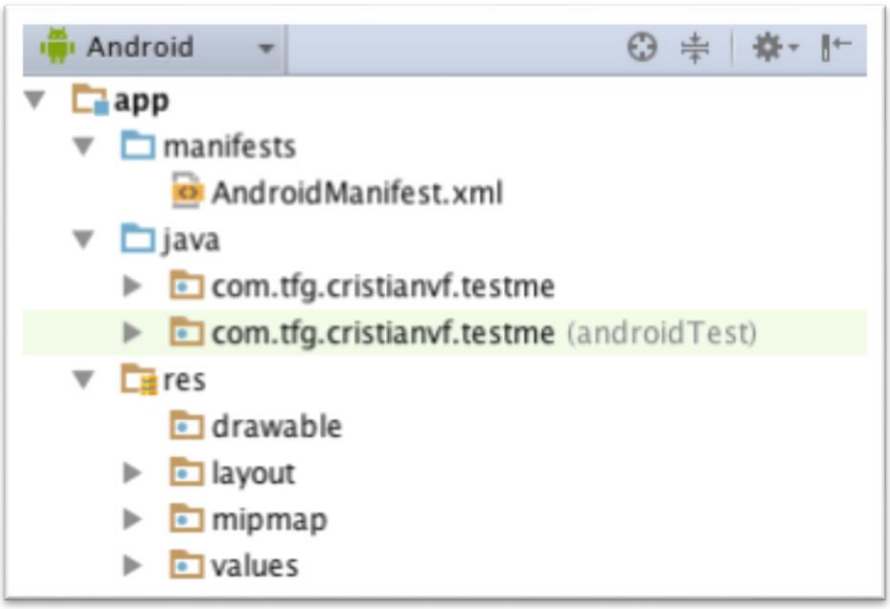

Figura 20 - Estructura de un proyecto en Android Studio.

3.2.4.3 Ejecutar un proyecto

Existen dos formas de ejecutar un proyecto en Android Studio:

- Mediante Android Virtual Device (AVD).
- Mediante un dispositivo físico.

Ejecución en Android Virtual Device (AVD)

Para ejecutar un proyecto de Android Studio en Android Virtual Device es necesario crear el el AVD y ejecutar el proyecto en él. Este proceso se detalla en los siguientes pasos.

1. Para crear un AVD se puede abrir el *AVD Manager* presente en la barra de herramientas o acceder al *AVD Manager* mediante *Tools* > *Android* > *AVD Manager*.
2. En el *AVD Manager* se selecciona la opción *Create Virtual Device*.
3. Como tercer paso, se selecciona un modelo de teléfono predefinido o se configura uno nuevo.
4. Se selecciona el nuevo AVD creado y se lanza pulsando Start.

5. Cuando el AVD ya está listo para ser utilizado, es turno de ejecutar el proyecto. Se puede ejecutar pulsando el icono *Run* de la barra de herramientas o mediante *Run* > *Run app*.
6. A continuación, se elige la opción "*Choose a running device*" y se selecciona el AVD.
7. Como último paso se pulsa Ok para instalar la aplicación en el AVD.

Ejecución en dispositivo físico

Para ejecutar un proyecto de Android Studio en un dispositivo físico es necesario configurar el teléfono y ejecutar el proyecto en él. Este proceso se detalla en los siguientes pasos.

1. En primer lugar, es necesario habilitar la depuración USB en el dispositivo. Esta opción se activa en *Ajustes* > *Opciones del desarrollador*.
2. Se conecta el dispositivo al ordenador a través de un cable USB.
3. El proyecto se ejecuta pulsando el icono *Run* de la barra de herramientas o mediante *Run* > *Run app*.
4. A continuación, se elige la opción "*Choose a running device*" y se selecciona el dispositivo.
5. Como último paso se pulsa Ok para instalar la aplicación en el dispositivo.

Depurar un proyecto

Android Studio incluye un depurador que permite depurar aplicaciones que se están ejecutando en un emulador de Android o en un dispositivo físico. Para realizar una depuración del código es necesario seguir los siguientes pasos:

1. Seleccionar el dispositivo o emulador en el que se va a realizar la depuración.
2. Establecer los *breakpoints* (puntos de parada) en el código.
3. La depuración comienza al pulsar el botón de *debug* en la barra de herramientas o accediendo a *Run* > *Debug app.*
4. Android Studio genera un APK, firmado con una clave de depuración, lo instala en el dispositivo o emulador y abre una ventana de depuración.
5. El *debugger* examina variables y evalúa expresiones en tiempo de ejecución.

3.3 Tecnologías utilizadas

3.3.1 HTTP

Hypertext Transfer Protocol (HTTP) es un protocolo de aplicación para la distribución e intercambio de información a través de la *World Wide Web*. HTTP actúa como un protocolo petición-respuesta en el modelo de computación cliente-servidor [33].

Hypertext es un texto estructurado que utiliza enlaces lógicos (*links*) entre nodos que contienen texto. La información a la que accede HTTP se identifica gracias a una URL (*Uniform Resource Locator*).

Petición

Un mensaje de petición HTTP consta de un método de petición, identificador de un recurso (URL) y la versión HTTP del cliente.

HTTP define una serie de métodos para realizar una determinada acción sobre un recurso concreto. Los métodos HTTP más utilizados son:

- GET: El método GET se utiliza para solicitar información o un recurso al servidor.
- POST: El método POST se utiliza para enviar datos al servidor y que éstos sean procesados por el recurso identificado.
- HEAD: El método HEAD se utiliza para obtener la cabecera de un determinado recurso, solicita el recurso de la misma manera que el método GET, pero no recibe el cuerpo del mensaje.
- PUT: El método PUT se utiliza para subir un recurso a un servidor.
- DELETE: El método DELETE se utiliza para borrar un recurso de un servidor.

Respuesta

Un mensaje de respuesta HTTP consta de la versión HTTP utilizada, un código de respuesta y un mensaje asociado a la respuesta.

Los códigos de estado de la respuesta indican qué ha sucedido con la petición que envió el cliente. Los códigos suelen presentar la siguiente disposición [33]:

- 1xx – Informativo: Estos códigos indican una respuesta provisional.
- 2xx – Éxito: Estos códigos indican que la petición del cliente fue recibida con éxito.
- 3xx – Redirección: Estos códigos indican que se necesita realizar alguna otra acción antes de realizar la petición. Por lo tanto, se redirige al cliente.
- 4xx – Error de cliente: Estos códigos indican que el cliente ha cometido algún error a la hora de realizar la petición.
- 5xx – Error de servidor: Estos códigos indican que ha ocurrido un error en la parte del servidor o que el servidor es incapaz de realizar la petición.

3.3.2 Java

Java es un lenguaje de programación de propósito general, orientado a objetos y concurrente desarrollado por Sun Microsystems en 1991.

La principal característica de Java es ser un lenguaje interpretado y compilado, es decir, todos los programas escritos en Java tienen que ser compilados. El código se compila y se generan *bytecodes* que se interpretan por la máquina virtual de Java (*Java virtual machine*). De esta manera, se consiguen programas que se ejecutan con independencia de la máquina en la que corran, porque el código se ejecuta en la máquina virtual de Java, que es independiente de la plataforma [34].

El SDK del sistema operativo Android (ver apartado 3.2.3.3) proporciona la posibilidad de desarrollar aplicaciones utilizando el lenguaje Java. Sin embargo, utiliza la máquina virtual Dalvik (*Dalvik virtual machine*) en lugar de utilizar la máquina de Java.

3.3.3 JSON

JSON (*JavaScript Object Notation*) es un formato ligero utilizado para el intercambio de datos. JSON es un formato de texto completamente independiente del lenguaje.

JSON está constituido por dos tipos de estructuras:

- Objeto JSON: Un objeto JSON es una colección de pares clave-valor. Cada uno de los pares se separa mediante comas, las claves-valores se asocian utilizando dos puntos y todo va entre llaves como se muestra en la Figura 21.

Figura 21 - Objeto JSON.

- Array JSON: Un array JSON es una lista que contienen objetos JSON. Cada uno de los objetos JSON se separa con comas. El principio y el fin del array se marca con corchetes (ver Figura 22).

Figura 22 - Array JSON.

Se utiliza JSON en el sistema para recibir los datos solicitados a la base de datos o la respuesta a una acción realizada sobre la de datos. Es el *web service* (ver apartado 4.12) quien actúa directamente con la base de datos, por lo tanto, los mensajes JSON son emitidos por el servidor web.

3.3.4 PHP

PHP (*PHP Hypertext Preprocessor*) es un lenguaje enfocado a la programación de scripts en el lado del servidor. El código de PHP se incrusta en código HTML y genera contenido dinámico. Su sintaxis es similar a Java, C y Perl [35].

PHP se ejecuta en el lado del servidor y el resultado se envía al cliente. Al ejecutarse en el lado del servidor, PHP tiene acceso a los recursos del servidor como la base de datos, realizando la conexión y autenticación correspondientes. Una de las principales características de PHP es que proporciona soporte para un gran número de bases de datos.

Asimismo, PHP funciona en la mayoría de servidores web y puede utilizarse en la mayoría de sistemas operativos.

Se utiliza PHP en el sistema desarrollado, mediante los ficheros PHP alojados en el servidor web, a modo de middleware para que realice las consultas a la base de datos que solicite la aplicación.

3.3.5 XML

XML (Extensible Markup Language) es un meta-lenguaje de etiquetado muy simple, que es parte fundamental en el intercambio de datos. Es similar a HTML, pero la función de XML es la de describir datos, mientras que la de HTML es la de mostrar datos. XML se utiliza para estructurar, almacenar e intercambiar información [36].

Se utiliza XML en la aplicación para definir los atributos y características principales de la interfaz de usuario en los archivos correspondientes a los *layout.* Asimismo, se utiliza XML para definir cadenas de texto, estilos y colores, en archivos XML distintos, que se utilizarán como parte de la interfaz de usuario. El propósito de definir estos atributos por separado es para mantener el código del layout lo más limpio y entendible posible, además de esta manera, estos atributos pueden ser reutilizados.

3.4 Software utilizado

3.4.1 Filezilla

Filezilla es una aplicación software FTP (*File Transfer Protocol*), es decir, su función es la de transferir archivos de un cliente a un servidor. Filezilla es multiplataforma, está disponible para Windows, Mac Os, Linux, etc.

Se ha elegido este cliente FTP para la transferencia de archivos PHP al servidor web, alojado en Hostinger (ver apartado 3.4.2), porque es software libre, es un cliente FTP seguro y está disponible para la mayoría de plataformas.

Para iniciar la transferencia de archivos, se tiene que realizar la conexión con el servidor web. Para ello, es necesario introducir los siguientes datos:

- Servidor: En este campo se tiene que identificar el servidor web con el se quiere realizar la conexión. En este caso, se ha utilizado el nombre de host FTP. También se puede utilizar la IP del servidor.
- Nombre de usuario: En este campo se tiene que especificar el nombre de usuario asociado al servidor web. En este caso, el nombre proporcionado por Hostinger.
- Contraseña: En este campo se tiene que introducir la contraseña del usuario asociado al servidor web con el se quiere realizar la conexión.
- Puerto: Se especifica el puerto por el cual se realiza la conexión. En este caso, se deja el puerto por defecto.

Figura 23 - Cliente FTP Filezilla.

3.4.2 Hostinger

Hostinger es un sitio web que ofrece servicios de *hosting* gratis y de pago. El plan de pago proporciona todos los servicios y características disponibles. Mientras que el plan gratuito solamente proporciona los servicios básicos.

Se ha elegido el plan de *hosting* gratis porque cubre las características necesarias para que el sistema que se está desarrollando funcione. El plan gratuito incluye las siguientes características [37]:

- 2000 MB de espacio en disco.
- 100 GB de ancho de banda.
- Soporte PHP.
- 2 bases de datos MySQL.
- Herramienta PHPMyAdmin.
- Acceso FTP.
- Usuario FTP.

3.4.3 MySQL

MySQL es un sistema gestor de bases de datos relacionales. Este sistema provee un servidor de base de datos SQL multihilo y multiusuario. MySQL es software de código abierto, su uso está muy extendido, existe mucha documentación para este sistema y se integra bien con PHP [38].

Se ha elegido MySQL porque ya se había trabajado con este sistema anteriormente, por lo que no es necesario aprender cómo utilizarlo y, además, es el sistema gestor de bases de datos que proporciona Hostinger.

En el sistema que se está desarrollando la función de MySQL es la de gestionar y almacenar la información relativa a los usuarios de la aplicación, los grados, las asignaturas, los temas y las preguntas en una base de datos.

3.4.4 PhpMyAdmin

PhpMyAdmin es una herramienta software gratis y de código libre escrita en PHP cuya principal tarea es administrar sistemas MySQL mediante un navegador web [39].

Las operaciones más comunes (administración de base de datos, tablas, columnas, etc) se pueden realizar mediante la interfaz de usuario o mediante línea de comandos.

Las características más importantes de phpMyAdmin son:

- Posee una interfaz web intuitiva.
- Soporte para la mayoría de funcionalidades de MySQL.
- Posibilidad de importar datos de archivos SQL y CSV.
- Crear esquemas visuales de las bases de datos.
- Multiplataforma.
- Multilenguaje, disponible en 72 idiomas.

En el sistema que se está desarrollando PhpMyAdmin permite administrar el sistema MySQL mediante una interfaz web de usuario, que facilita en gran manera las acciones realizadas sobre la base de datos, como creación de tablas, inserción de datos, etc. Además, permite realizar *backups* de la base de datos. En la Figura 24 se muestra la vista principal de phpMyAdmin, que contiene la estructura de la base de datos.

Figura 24 - phpMyAdmin.

3.4.5 Postman

Postman es una herramienta software distribuida como aplicación para Windows y Mac, y como plugin para Google Chrome. Postman permite gestionar de manera sencilla peticiones a servicios web REST.

El usuario de Postman construye una petición a un servicio web REST y la envía al servidor. La petición se procesa en el servidor y la respuesta es capturada e interpretada por esta herramienta. Entonces, se muestra la respuesta con el formato JSON, HTML, XML o texto plano [40].

Se ha utilizado Postman, en este proyecto, para realizar pruebas sobre el correcto funcionamiento de las peticiones al servicio web REST que se realizan desde la aplicación para acceder a los datos almacenados en la base de datos mediante los recursos PHP alojados en el servidor web.

La Figura 25 muestra la pantalla principal de Postman con una petición GET realizada y su respectiva respuesta.

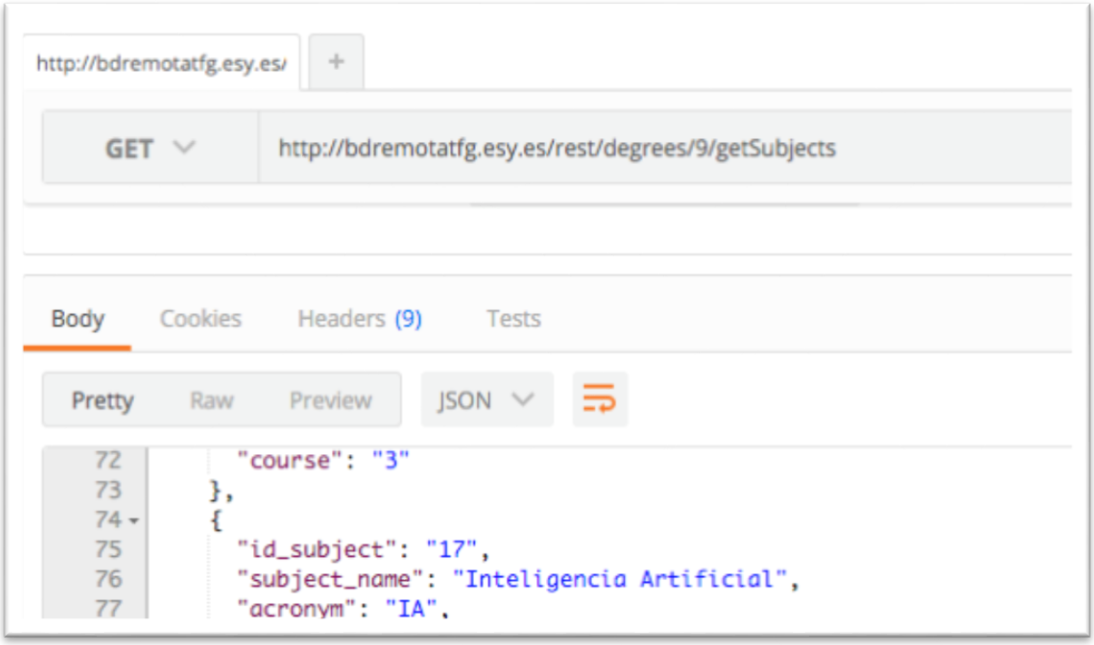

Figura 25 - Postman.

3.4.6 Gantt Project

GanttProject es una aplicación de escritorio multiplataforma para la programación y gestión de proyectos. Es software libre, posee licencia GPL, y su código es opensource. Está disponible para Windows y MacOsX [41].

Su principal funcionalidad es la creación de diagramas de Gantt. El diseño de estos diagraas permite visualizar el desarrollo de un proyecto desglosado en tareas, eventos o hitos, lo cual permite gestionar la planificación temporal de los proyectos.

3.4.7 Librería Volley

Volley es una librería HTTP que facilita la creación de conexiones para aplicaciones Android, de una manera más rápida [42]. Se ha utilizado Volley en el proyecto para realizar la conexión remota de la aplicación con el servidor.

Volley proporciona una manera más sencilla y limpia de trabajar con conexiones que la que ofrece la clase AsyncTask. A la hora de realizar funciones que, debido a la conexión pueden tardar un tiempo en responder, es necesario realizarlas en segundo plano. Para ello, Android facilita la clase AsyncTask, que permite que se programen trozos de código que se ejecutarán específicamente en segundo plano, siendo necesario poner este código en una clase de Java aparte. Mientras que Volley, realiza la creación de hilos que se ejecutarán en segundo plano automáticamente, ahorrándo trabajo y líneas de código.

Para incluir Volley al proyecto Android solamente es necesario incluir la siguiente línea en las dependencias de proyecto:

```
compile 'com.android.volley:volley:1.0.0'
```

3.5 Especificación de requisitos

En este punto se realiza la especificación de los requisitos del sistema. La finalidad de estos requisitos es servir como guía para la fase de desarrollo de la aplicación y para el diseño de la misma. Asimismo, estos requisitos se utilizan para validar el sistema durante la fase de evaluación.

Se han utilizado dos tipos de requisitos para este apartado. Los requisitos funcionales y los no funcionales.

- Los requisitos funcionales definen las funcionalidades de debe cumplir la aplicación.
- Los requisitos no funcionales definen las limitaciones del sistema, es decir, establecen los atributos o propiedades del sistema.

Los requisitos se especifican siguiendo la estructura definida en la Tabla 4:

Identificador del requisito			
Nombre	Nombre del requisito.		
Dependencias	Requisitos de los que depende.	Prioridad	Alta, media o baja.
Descripción	Breve descripción del requisito.		

Tabla 4. Formato de los requisitos

- El campo de identificador de requisitos se utiliza para identificar a cada uno de los requisitos. Este identificador sigue el formato RX-YY. Donde X expresa el tipo de requisito, es decir, indica si funcional o no funcional mediante ‘F’ y ‘NF’, respectivamente. Mientras que YY indica el número del requisito, que comienza a partir del 00.
- El campo nombre se utiliza para dar una idea del objetivo del requisito.
- Las dependencias indican los requisitos de los que depende este requisito mediante sus identificadores.
- La prioridad representa el grado de necesidad de incluir dicha funcionalidad o propiedad al sistema. La prioridad puede ser alta, media o baja.
- Por último, en el campo descripción se realiza una explicación breve y concisa del requisito.

3.5.1 Requisitos funcionales

RF-01			
Nombre	Registro de grados.		
Dependencias	-	Prioridad	Alta
Descripción	Durante el registro, la aplicación debe ofrecer todos los grados correspondientes a la rama de conocimiento elegida por el usuario.		

Tabla 5. RF-01: Registro de grados.

RF-02			
Nombre	Primera matriculación de asignaturas.		
Dependencias	RF-01	Prioridad	Media
Descripción	La primera vez que se inicia sesión, la aplicación debe mostrar una pantalla con todas las asignaturas disponibles a matricular. Estas asignaturas pertenecen al grado elegido al realizar el registro.		

Tabla 6. RF-02: Primera matriculación de asignaturas.

RF-03			
Nombre	Asignaturas matriculadas.		
Dependencias	RF-02, RF-04 y RF-05	Prioridad	Alta
Descripción	La aplicación debe mostrar las asignaturas matriculadas por el usuario en la pantalla principal.		

Tabla 7. RF-03: Asignaturas matriculadas.

RF-04			
Nombre	Añadir asignaturas.		
Dependencias	-	Prioridad	Alta
Descripción	La aplicación debe permitir añadir asignaturas a la lista de asignaturas matriculadas.		

Tabla 8. RF-04: Añadir asignaturas.

RF-05			
Nombre	Eliminar asignaturas.		
Dependencias	-	Prioridad	Alta
Descripción	La aplicación debe permitir eliminar asignaturas de la lista de asignaturas matriculadas.		

Tabla 9. RF-05: Eliminar asignaturas.

RF-06			
Nombre	Menú de opciones		
Dependencias	-	Prioridad	Alta
Descripción	La aplicación tiene que ofrecer un menú que permita editar los datos del usuario, ver estadísticas de la realización de test y cerrar sesión.		

Tabla 10. RF-06: Menú de opciones.

RF-07			
Nombre	Estadísticas de asignaturas		
Dependencias	RF-06	Prioridad	Media
Descripción	La aplicación tiene que proporcionar información acerca de los resultados obtenidos en los test realizados para cada asignatura. Esta funcionalidad es una opción del menú de la pantalla principal.		

Tabla 11. RF-07: Estadísticas de asignaturas.

RF-08			
Nombre	Estadísticas de temas		
Dependencias	RF-06	Prioridad	Media
Descripción	La aplicación tiene que proporcionar información acerca de los resultados obtenidos en los test realizados desglosados por temas. Esta funcionalidad es una opción del menú de la pantalla de temario.		

Tabla 12. RF-08: Estadísticas de temas.

RF-09			
Nombre	Mostrar temario		
Dependencias	RF-03	Prioridad	Alta
Descripción	El sistema tiene que mostrar la lista de temas que pertenecen a la asignatura elegida en la pantalla principal.		

Tabla 13. RF-09: Mostrar temario.

RF-10			
Nombre	Respuesta multimodal.		
Dependencias	-	Prioridad	Alta
Descripción	Se proporcionan una pregunta y sus cuatro posibles respuestas. El sistema debe permitir al usuario responder mediante voz y mediante la pantalla táctil.		

Tabla 14. RF-10: Respuesta multimodal.

RF-11			
Nombre	Feedback.		
Dependencias	RF-10	Prioridad	Alta
Descripción	La aplicación tiene que dar retroalimentación al usuario cuando contesta a una pregunta. El *feedback* puede ser visual o mediante un mensaje con texto.		

Tabla 15. RF-11: Feedback.

RF-12			
Nombre	Resultados		
Dependencias	-	Prioridad	Alta
Descripción	La aplicación debe llevar un recuento sobre las respuestas acertadas y las respuestas fallidas para mostrar el resultado obtenido al finalizar el test.		

Tabla 16. RF-12: Resultados

RF-13			
Nombre	Editar grado.		
Dependencias	-	Prioridad	Media
Descripción	El sistema debe permitir al usuario editar el grado inscrito, cambiando también el listado de asignaturas disponibles.		

Tabla 17. RF-13: Editar grado.

RF-14			
Nombre	Editar datos de usuario		
Dependencias	-	Prioridad	Baja
Descripción	El sistema debe permitir al usuario cambiar su nombre de usuario y su contraseña.		

Tabla 18. RF-14: Editar datos de usuario.

RF-15			
Nombre	Cerrar sesión.		
Dependencias	-	Prioridad	Media
Descripción	La aplicación debe permitir al usuario cerrar su sesión.		

Tabla 19. RF-15: Cerrar sesión.

RF-16			
Nombre	Inicio en pantalla principal.		
Dependencias	RF-15	Prioridad	Baja
Descripción	La aplicación debe permitir al usuario iniciar la aplicación en la pantalla principal si no se cerró la sesión la última vez que el usuario quitó la aplicación.		

Tabla 20. RF-16: Inicio en pantalla principal.

RF-17			
Nombre	Navegación atrás.		
Dependencias	-	Prioridad	Baja
Descripción	La aplicación debe permitir la navegación atrás, hasta llegar a las pantallas de login o pantalla principal.		

Tabla 21. RF-17: Navegación atrás.

RF-18			
Nombre	Salir de la aplicación.		
Dependencias	RF-17	Prioridad	Baja
Descripción	Al realizar navegación atrás en las pantallas de login o principal, la aplicación debe mostrar un mensaje de confirmación para cerrar la aplicación. Y en caso afirmativo, cerrar la aplicación		

Tabla 22. RF-18: Salir de la aplicación

3.5.2 Requisitos no funcionales

RNF-01			
Nombre	Breve tiempo de respuesta.		
Dependencias	-	Prioridad	Alta
Descripción	La aplicación tiene que responder a las acciones del usuario de manera instantánea.		

Tabla 23. RNF-01: Breve tiempo de respuesta.

RNF-02			
Nombre	Formato de los recursos externos.		
Dependencias	-	Prioridad	Media
Descripción	El formato de los recursos externos utilizados por la aplicación como imágenes o sonidos, deben ser compatibles con Android.		

Tabla 24. RNF-02: Formato de los recursos externos.

RNF-03			
Nombre	Navegación de la interfaz.		
Dependencias	RNF-01	Prioridad	Alta
Descripción	La navegación entre las diferentes pantallas que componen la aplicación debe ser fluido.		

Tabla 25. RNF-03: Navegación de la interfaz.

RNF-04			
Nombre	Resolución de la interfaz.		
Dependencias	-	Prioridad	Media
Descripción	La interfaz de usuario tiene que adaptarse a distintos tamaños de pantalla y diferentes resoluciones de manera automática.		

Tabla 26. RNF-04: Resolución de la interfaz.

RNF-05			
Nombre	Compatibilidad de versiones.		
Dependencias	-	Prioridad	Alta
Descripción	La aplicación deber ser compatible con la versión 4.4 de Android y las versiones superiores.		

Tabla 27. RNF-05: Compatibilidad de versiones

Capítulo 4

4. Diseño y desarrollo de la aplicación

En este capítulo se va a detallar el diseño de la interfaz de usuario, así como el desarrollo y funcionalidad de la aplicación y sus componentes.

Las principales características que debe cumplir la interfaz de la aplicación son las siguientes:

- Sencillez: La interfaz será intuitiva y minimalista para simplificar su uso.
- Claridad: La interfaz debe ser perfectamente entendible para el usuario.
- Familiaridad: Se utilizarán conceptos conocidos por los usuarios.
- Consistencia: Las acciones y conceptos similares tendrán una representación y disposición similar en la interfaz.
- Comportamiento fiable: El comportamiento de la interfaz debe ser siempre el esperado, de manera que no cause pérdida de información o malfuncionamiento de la aplicación.
- Fácil acceso: Todo el contenido debe ser accesible sin necesidad de recorrer complicados o largos caminos.

Para cada una de las pantallas se ha realizado su correspondiente *mockup* dibujado a mano, que ha sido utilizado como base para construir la interfaz de la aplicación.

Seguidamente, se va a detallar el diseño final de la interfaz de usuario de la aplicación a partir de los bocetos. El diseño final de la interfaz de usuario se ha realizado utilizando Android Studio.

Asimismo, se explicarán dificultades encontradas a la hora de llevar los diseños del *mockup* a su fase funcional y los elementos nuevos incorporados para satisfacer las funcionalidades añadidas a medida que se iba avanzando en el desarrollo de la aplicación.

4.1 Login

4.1.1 Diseño inicial de la pantalla de login

La Figura 26 muestra el diseño de la pantalla de inicio de sesión. El objetivo de esta pantalla es acceder a la aplicación con los datos de un usuario previamente registrado, para ello se ha incluido los siguientes elementos de interfaz:

- Dos cuadros de texto donde se introducirán el nombre de usuario y la contraseña, respectivamente.
- Un botón que permitirá iniciar sesión.
- Un botón que permitirá acceder a la sección de registro de usuario.

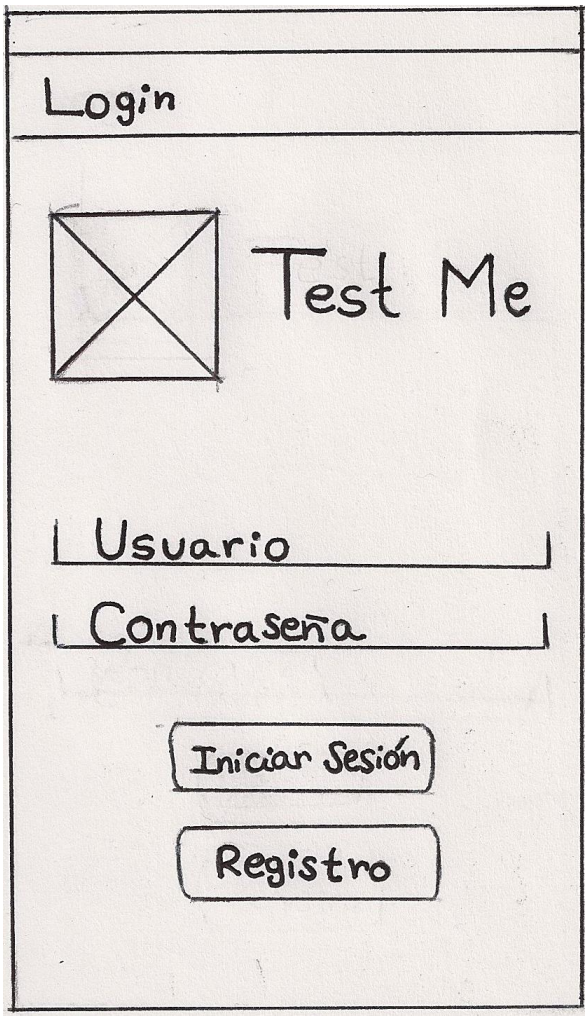

Figura 26 - Login. Mockup.

4.1.2 Diseño final de la pantalla de login

La pantalla de login, Figura 27, se ha modelado siguiendo el diseño presentado en el apartado 4.1.1. Al diseño inicial se ha añadido un *TextView* clickable que encamina a una pantalla para recuperar la contraseña. El resto de elementos se han construido con dos *EditText* para introducir el nombre de usuario y la contraseña, respectivamente; y dos botones para iniciar sesión y pasar a la pantalla de registro, respectivamente.

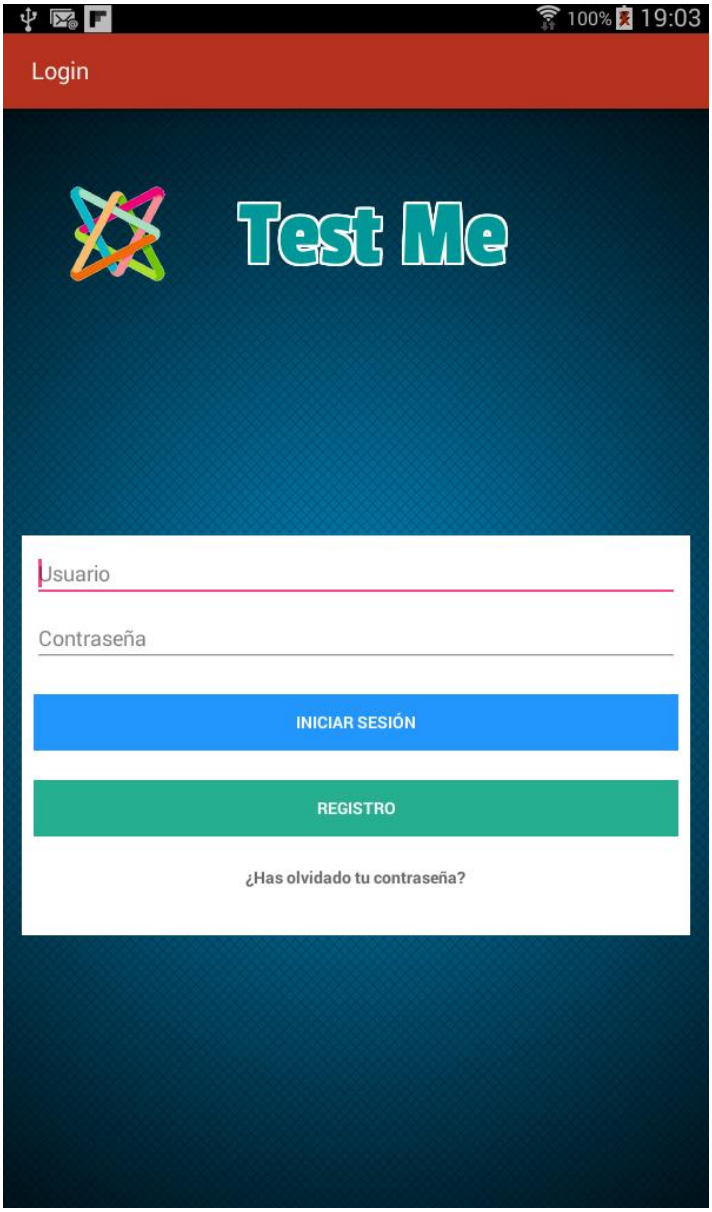

Figura 27 - Login. Diseño final.

4.1.3 Funcionalidad

El usuario puede iniciar sesión identificándose mediante su nombre de usuario y contraseña, que deben haber sido previamente registrados. En caso de no estar registrado, el usuario deberá registrarse en la aplicación. El usuario puede acceder al módulo de registro mediante el botón de registro. Asimismo, el usuario puede pasar al módulo de recuperación de contraseña mediante el enlace correspondiente. Los elementos referidos se presentan en la Figura 27.

Cuando el usuario ha iniciado sesión correctamente se pueden dar tres casos:

- Es la primera vez que el usuario inicia sesión, entonces, se accede a la *activity* de inscripción de asignaturas para que el usuario seleccione las asignaturas que desee estudiar.
- El usuario ha eliminado todas sus asignaturas inscritas y ha cerrado sesión. Al volver a iniciar sesión se da la posibilidad al usuario de matricular asignaturas nuevamente accediendo a la *activity* de inscripción de asignaturas.
- El usuario ya ha accedido previamente a la aplicación y tiene asignaturas matriculadas, entonces, se cargan los datos del usuario desde la base de datos y se almacenan en la aplicación mediante *sharedPreferences* para su posterior uso en otras *activities*. Además, se guarda un dato extra, que permite mantener iniciada la sesión aunque el usuario cierre la aplicación.

4.1.4 Flujo de datos

Para la implementación del inicio de sesión se ha utilizado una arquitectura cliente-servidor. En el lado del servidor se ha utilizado el lenguaje PHP y la base de datos remota MySQL. El lado del cliente (aplicación Android) se comunica con el servidor por medio de HTTP y JSON.

El proceso de comunicación entre el cliente y el servidor se explica en los siguientes puntos. Asimismo, se muestra una representación visual del intercambio de datos, que se genera al iniciar sesión, en la Figura 28.

1. El usuario introduce su nombre de usuario y contraseña con el propósito de acceder a la aplicación.

2. La aplicación envía una petición GET al servicio web RESTful especificado en el apartado 4.12, con nombre de usuario y contraseña como parámetros.
3. El servicio web ejecuta el código PHP y envía una consulta a la tabla **User** de la base de datos que comprueba que existe un usuario cuyo nombre de usuario y contraseña coincidan con los parámetros enviados por el usuario.
4. La base de datos devuelve los campos de la fila asociada al nombre de usuario y contraseña.
5. El servidor PHP obtiene los datos devueltos por la base de datos, los convierte a formato JSON y los envía a la aplicación.
6. La aplicación recibe los datos y los almacena en SharedPreference.
7. Se inicia la *activity* principal o la *activity* de inscripción de asignaturas (ver apartado 4.1.3).

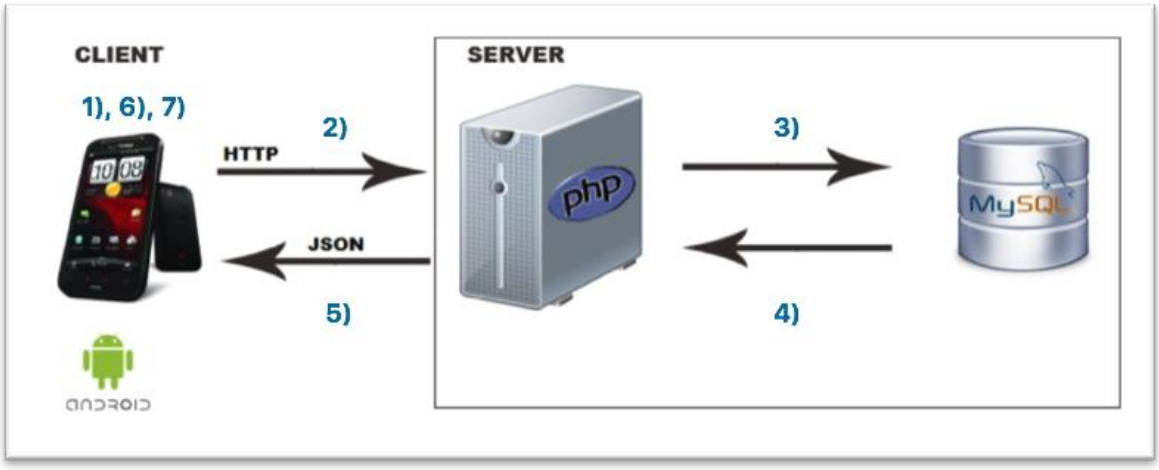

Figura 28 - Login. Flujo de datos.

4.2 Registro

4.2.1 Diseño inicial de la pantalla de registro

La Figura 29 muestra el diseño inicial de la pantalla de registro. El objetivo de esta pantalla es permitir a un usuario registrarse en la aplicación con los datos que proporcione, para ello se ha incluido los siguientes elementos de interfaz:

- Un cuadro de texto para introducir el nombre de usuario.
- Un cuadro de texto para introducir la dirección de correo electrónico.
- Dos cuadros de texto para introducir la contraseña.
- Un cuadro de selección de la rama de conocimiento.
- Un cuadro de selección del grado a matricular.
- Un botón para confirmar el registro.

Figura 29 - Registro. Mockup.

4.2.2 Diseño final de la pantalla de registro

La pantalla de registro, Figura 30, se ha modelado siguiendo el diseño presentado en el apartado 4.2.1. El diseño final se ha construido con cuatro *EditText* para introducir el nombre de usuario, la dirección de correo electrónico, la contraseña y la confirmación de la contraseña, respectivamente. Dos *Spinner* para elegir la rama de conocimiento y el grado que el usuario desee matricular, respectivamente. Y un botón para registrar los datos introducidos por el usuario, guardarlos en la base de datos y volver a la pantalla de login.

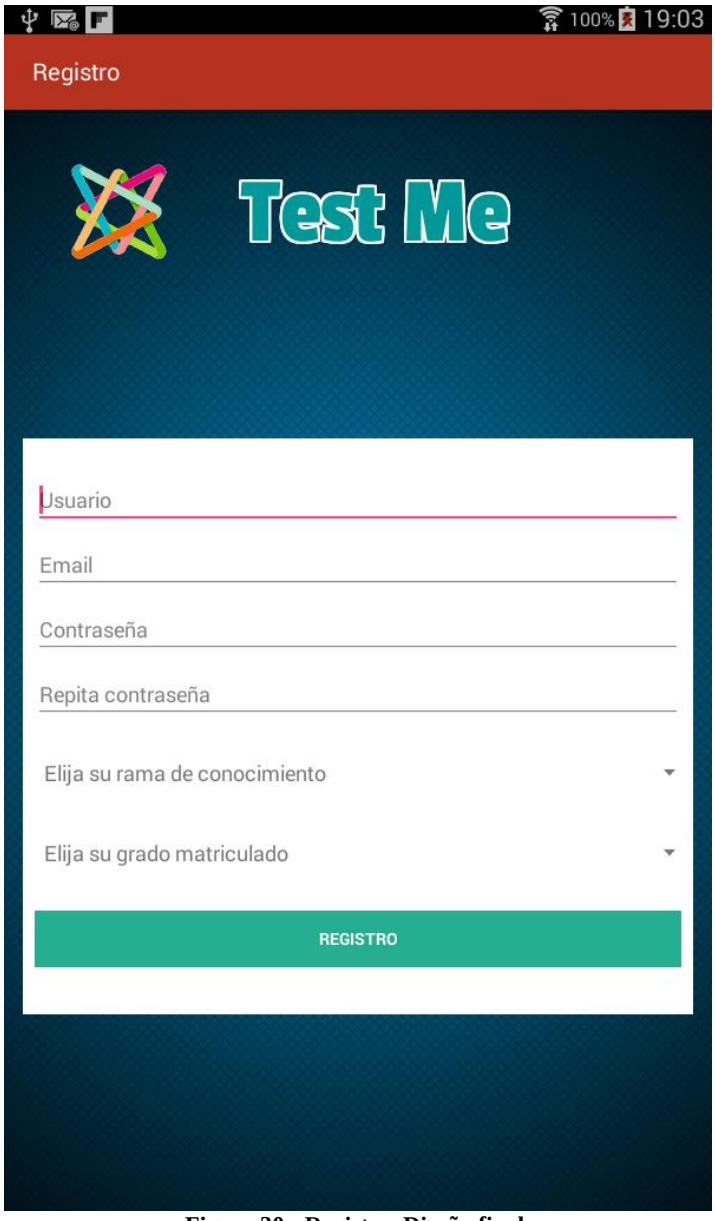

Figura 30 - Registro. Diseño final.

4.2.3 Funcionalidad

El usuario puede registrar su nombre de usuario y contraseña en la aplicación para posteriormente acceder a los contenidos de la aplicación.

El usuario registra sus datos rellenando el formulario de registro que se muestra en la Figura 30. La aplicación envía la información a la base de datos remota MySQL donde los datos se harán persistentes.

Los datos que registra el usuario son los siguientes:

- El nombre usuario, email y contraseña del usuario se introducen rellenando los *EditText.*
- Para seleccionar un grado, el usuario debe elegir, en primer lugar, una de las ramas de conocimientos disponibles en el *Spinner* correspondiente. Una vez seleccionada la rama de conocimiento se cargan los grados pertenecientes a dicha rama. Entonces, el usuario selecciona el grado que desea matricular del *Spinner* correspondiente.

4.2.4 Flujo de datos

Para la implementación del registro se ha utilizado una arquitectura cliente-servidor. En el lado del servidor se ha utilizado el lenguaje PHP y la base de datos remota MySQL. El lado del cliente (aplicación Android) se comunica con el servidor por medio de HTTP y JSON.

El proceso de comunicación entre el cliente y el servidor se explica en los siguientes puntos. Asimismo, se muestra una representación visual del intercambio de datos, que se genera al registrar un usuario, en la Figura 31.

1. El usuario introduce sus datos: nombre de usuario, email, contraseña y rama de conocimiento.
2. La aplicación envía una petición GET al servicio web REST, especificado en el apartado 4.12, con la rama de conocimiento como parámetro.
3. El servicio web ejecuta el código PHP y envía una consulta a la tabla **Degree** de la base de datos que selecciona todos los grados cuyo campo “rama de conocimiento” se corresponda con el valor de la rama enviada por parámetro.
4. La base de datos devuelve los nombres y los identificadores de las filas asociadas a los grados resultantes de la consulta SQL.

5. El servidor web obtiene los datos devueltos por la base de datos, los convierte a formato JSON y los envía a la aplicación.
6. La aplicación recibe los datos relativos a los grados y los distribuye en el Spinner correspondiente a los grados.
7. Una vez seleccionado el último dato a registrar (grado a matricular), el usuario presiona el botón de registro.
8. La aplicación envía una petición GET al servicio web REST con el nombre de usuario y el email como parámetros.
9. El servicio web ejecuta el código PHP y envía una consulta a la tabla **User** de la base de datos que comprueba si el usuario o el email han sido registrados por otro usuario.
10. La base de datos devuelve el identificador del usuario si el email o el nombre de usuario están en uso.
11. El servidor PHP envía un código numérico, que indica si los datos del usuario están en uso, al cliente en formato JSON.
12. La aplicación recibe el código. Si el usuario o email no están registrados se prosigue con el registro. En caso contrario, se notifica al usuario que los datos ya están en uso.
13. Continuando con el registro, la aplicación envía una petición POST al servicio web REST con los datos de registro, es decir, nombre de usuario, email, contraseña y grado.
14. El servicio web ejecuta el código PHP y envía una consulta a la base de datos para registrar un nuevo usuario.
15. En la base de datos, se crea una nueva entrada en la tabla **User** con los datos introducidos por el usuario.
16. Una vez completado el registro, se inicia la *activity* login.

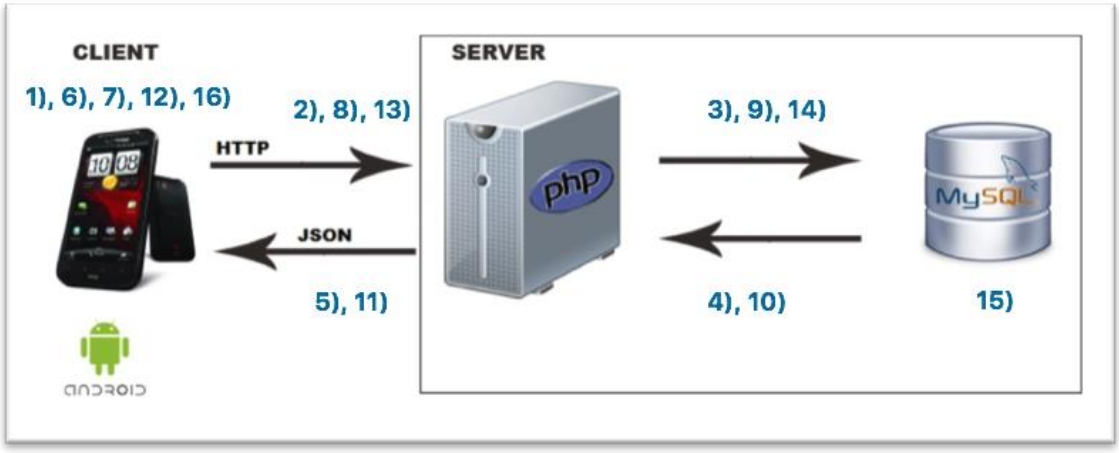

Figura 31 - Registro. Flujo de datos.

4.3 Recuperar contraseña

4.3.1 Diseño de la pantalla de recuperación de contraseña

La pantalla para recuperar la contraseña, Figura 32, se ha introducido para añadir una nueva funcionalidad que permita al usuario recuperar su clave en caso de olvido o pérdida de la misma. El diseño se ha realizado con un *EditText* para introducir la dirección de correo electrónico asociada a la cuenta cuya contraseña se quiera rescatar. Y un botón para validar la dirección email a la que se enviará la nueva contraseña para poder acceder a la cuenta solicitada.

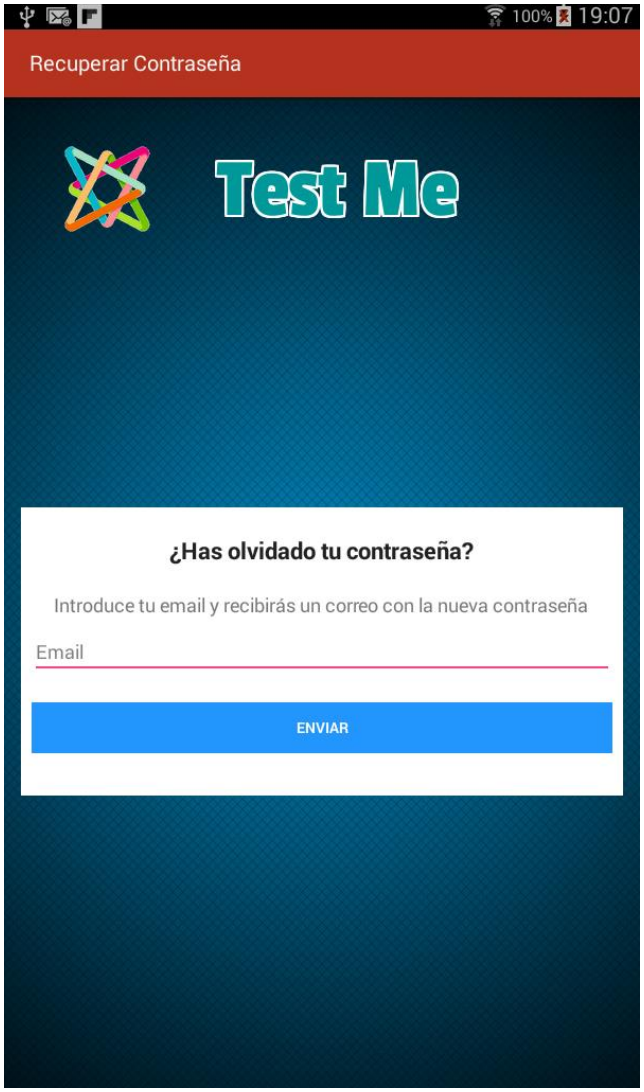

Figura 32 - Recuperar contraseña. Diseño final.

4.3.3 Funcionalidad

El usuario puede recuperar la contraseña de acceso a la aplicación mediante un correo electrónico enviado a la dirección email introducida por el usuario. La dirección email especificada debe estar registrada en la base de datos para que la nueva contraseña sea enviada a dicha dirección de correo electrónico.

El usuario introduce su dirección de correo electrónico en el campo de texto que se muestra en la Figura 32. Entonces, se genera una nueva contraseña que es enviada a la dirección determinada por el usuario. De esta manera, el usuario puede volver a acceder a la aplicación con su cuenta y restablecer la contraseña.

4.3.4 Flujo de datos

Para la implementación de la restauración de contraseña se ha utilizado una arquitectura cliente-servidor. En el lado del servidor se ha utilizado el lenguaje PHP y la base de datos remota MySQL. El lado del cliente (aplicación Android) se comunica con el servidor por medio de HTTP y JSON.

El proceso de comunicación entre el cliente y el servidor se explica en los siguientes puntos. Asimismo, se muestra una representación visual del intercambio de datos, que se genera al recuperar una contraseña, en la Figura 33.

1. El usuario introduce la dirección de correo electrónico donde recibirá la nueva contraseña.
2. La aplicación envía una petición GET al servicio web RESTful, especificado en el apartado 4.12, con el email como parámetro.
3. El servicio web ejecuta el código PHP y envía una consulta a la tabla **User** de la base de datos que comprueba si existe un usuario cuyo email se corresponda con el email pasado por parámetro.
4. La base de datos devuelve el identificador de la fila asociada al usuario resultante de la sentencia SQL.
5. El servidor web envía un código numérico, que indica si la dirección email está registrada, al cliente en formato JSON.
6. La aplicación recibe los datos, si la dirección email está asociada a un usuario registrado, se genera una nueva contraseña y la aplicación envía un mensaje con la nueva contraseña a la dirección email. En caso de que el email no exista en la base de datos, se notifica al usuario.

7. La aplicación envía una petición POST al servicio web RESTful con los datos de la contraseña recién generada.
8. El servicio web ejecuta el código PHP y envía una sentencia a la base datos para actualizar el usuario.
9. En la base de datos MySQL, se actualiza la contraseña del usuario guardando la nueva contraseña
10. Una vez completada la restauración de la contraseña, se inicia la *activity* login.

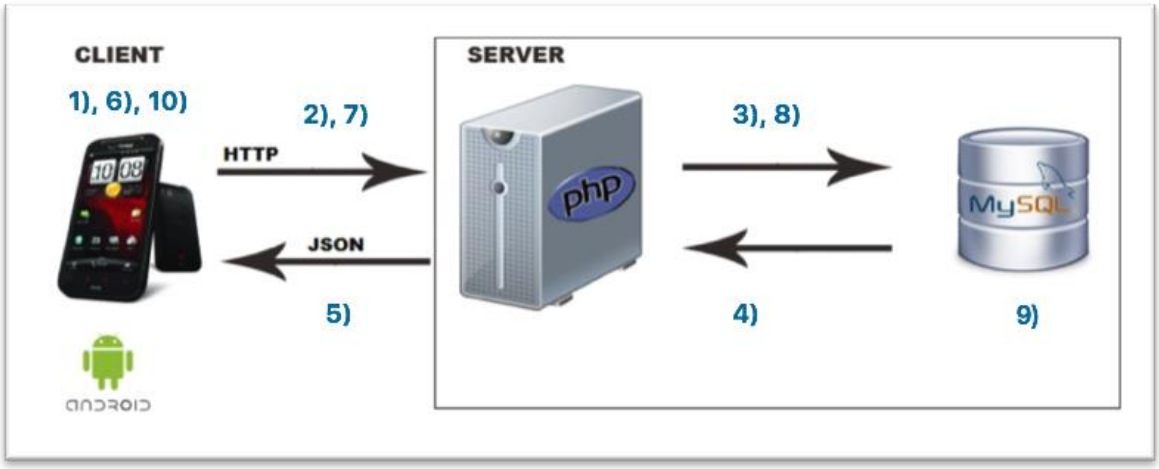

Figura 33 - Recuperar contraseña. Flujo de datos.

4.4 Inscripción de asignaturas

4.4.1 Diseño inicial de la pantalla de inscripción de asignaturas

La Figura 34 muestra el diseño inicial de la pantalla de inscripción de asignaturas. El objetivo de esta pantalla es mostrar todas las asignaturas pertenecientes a un grado en una lista y permite al usuario seleccionar las asignaturas que desee inscribir. Para ello se ha incluido los siguientes elementos de interfaz:

- Una lista que contiene todas las asignaturas disponibles en un grado, elegido al registrarse.
    - Cada elemento de la lista se compone de una imagen, el nombre de la asignatura y un checkbox que indica las asignaturas seleccionadas.
- Un botón para confirmar el registro.

Figura 34 - Inscripción de asignaturas. Mockup.

4.4.2 Diseño final de la pantalla de inscripción de asignaturas

La pantalla de inscripción de asignaturas, Figura 35, se ha modelado siguiendo el diseño presentado en el apartado 4.4.1. El diseño final se ha construido con un *ListView* que contiene todas las asignaturas del grado elegido al registrarse. Y un botón que asociará todas las asignaturas, seleccionadas mediante los *checkbox* de la lista, al usuario que las seleccionó y guarda los datos relativos a ambos en la base de datos.

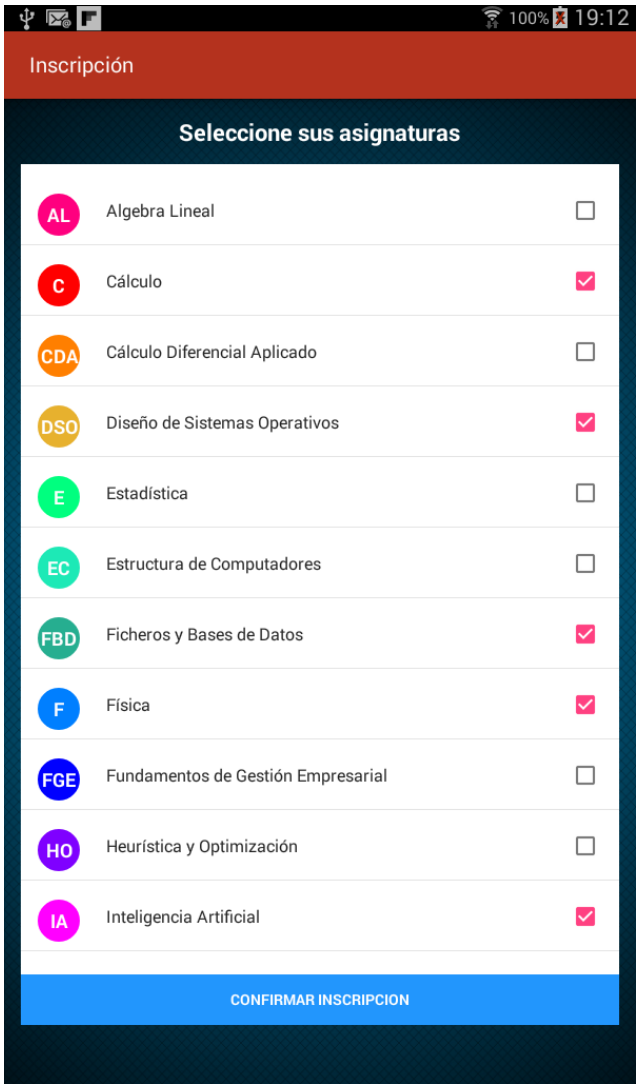

Figura 35 - Inscripción de asignaturas. Diseño final.

4.4.3 Funcionalidad

Para acceder a esta pantalla se tiene que dar una de estas situaciones:

- Es la primera vez que el usuario inicia sesión en la aplicación. Entonces, se accede a esta sección, a modo de ayuda, para que el usuario seleccione las asignaturas que desea inscribir.
- El usuario ha eliminado todas las asignaturas que tenía inscritas, ha cerrado sesión y minutos o días después inicia sesión.

Una vez en el módulo de inscripción de asignaturas (Figura 35) se cargan los datos del usuario almacenados en *SharedPreferences*. En este caso, se utiliza el identificador del grado, matriculado por el usuario, para recuperar las asignaturas pertenecientes al grado de la base de datos. Se obtiene una colección de asignaturas que se distribuyen en un *ListView*, cada asignatura se coloca en una fila de la lista. Asimismo, se coloca una imagen y un checkbox acompañando al nombre de la asignatura en cada una de las filas.

El usuario selecciona las asignaturas, que quiera matricular, activando los *checkbox* correspondientes, a continuación, se asocian estas asignaturas al usuario en la base de datos.

4.4.4 Flujo de datos

Para la implementación de la inscripción de asignaturas se ha utilizado una arquitectura cliente-servidor. En el lado del servidor se ha utilizado el lenguaje PHP y la base de datos remota MySQL. El lado del cliente (aplicación Android) se comunica con el servidor por medio de HTTP y JSON.

El proceso de comunicación entre el cliente y el servidor se explica en los siguientes puntos. Asimismo, se muestra una representación visual del intercambio de datos, que se genera al inscribir asignaturas, en la Figura 36.

1. La aplicación obtiene el identificador del grado matriculado por el usuario, dato que está almacenado en *SharedPreferences*.
2. La aplicación envía una petición GET al servicio web RESTful, especificado en el apartado 4.12, con el identificador del grado como parámetro.

3. El servicio web ejecuta el código PHP y envía una consulta a la tabla **Subject** de la base de datos que selecciona todas las asignaturas relativas al grado.
4. La base de datos devuelve los datos de la fila asociada a las asignaturas resultantes de la consulta SQL.
5. El servidor PHP envía los datos, identificador y nombre de asignaturas, al cliente en formato JSON.
6. La aplicación recibe los datos, y coloca los datos de las asignaturas en una lista.
7. El usuario selecciona las asignaturas que desee matricular y confirma las suscripciones.
8. La aplicación envía tantas peticiones POST como asignaturas haya seleccionado el usuario al servicio web RESTful.
9. A cada petición POST se le añaden los parámetros: identificador de usuario e identificador de asignatura.
10. El servicio web ejecuta el código PHP y envía sentencias a la base datos para matricular al usuario en las asignaturas.
11. En la base de datos MySQL, se crea una nueva entrada en la tabla **User_Subject** con los datos de usuario y de asignatura.
12. Una vez completada la inscripción, se inicia la *Activity* principal.

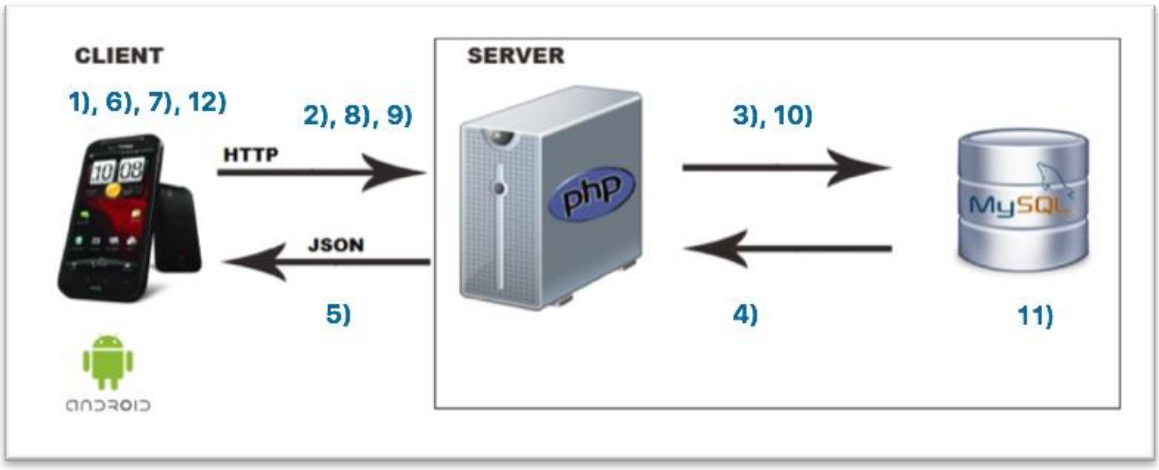

Figura 36 - Inscripción de asignaturas. Flujo de datos.

4.5 Pantalla principal

4.5.1 Diseño inicial de la pantalla principal

La Figura 37 muestra el diseño inicial de la pantalla principal. El propósito de esta pantalla es mostrar todas las asignaturas inscritas, en una lista y permite al usuario acceder al contenido de cada una de las asignaturas. Para ello se ha incluido los siguientes elementos de interfaz:

- Una lista que contiene todas las asignaturas elegidas en la inscripción.
    - Cada elemento de la lista se compone de una imagen, el nombre de la asignatura, el nombre del grado y el curso al que pertenece.
- Un botón para acceder a la inscripción de nuevas asignaturas.
- Un botón para acceder a la sección de quitar asignaturas inscritas.
- Un botón que despliega el menú de opciones.

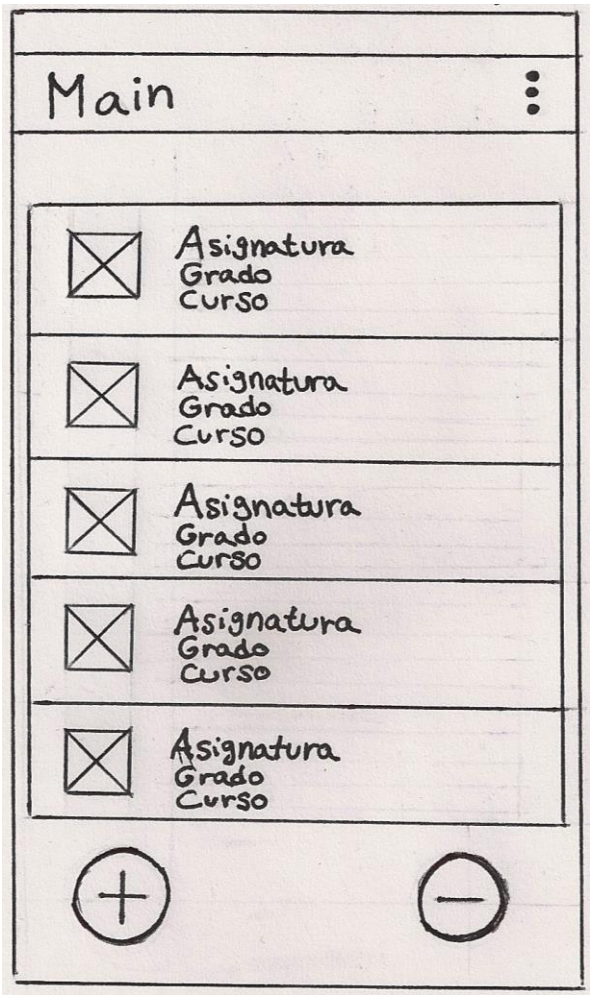

Figura 37 - Home. Mockup.

4.5.2 Pantalla principal

La pantalla principal o pantalla de asignaturas, Figura 38, se ha modelado siguiendo el diseño presentado en el apartado 4.5.1. El diseño final se ha construido con un *ListView* que contiene todas las asignaturas inscritas por el usuario. Dos *Floating Action Button* cuyo objetivo es añadir asignaturas y eliminar asignaturas, respectivamente.

Además, se han incluido el menú en el *Action Bar*, en caso de no haber espacio suficiente para distribuir los elementos del menú en el *Action Bar* se establece automáticamente un menú desplegable.

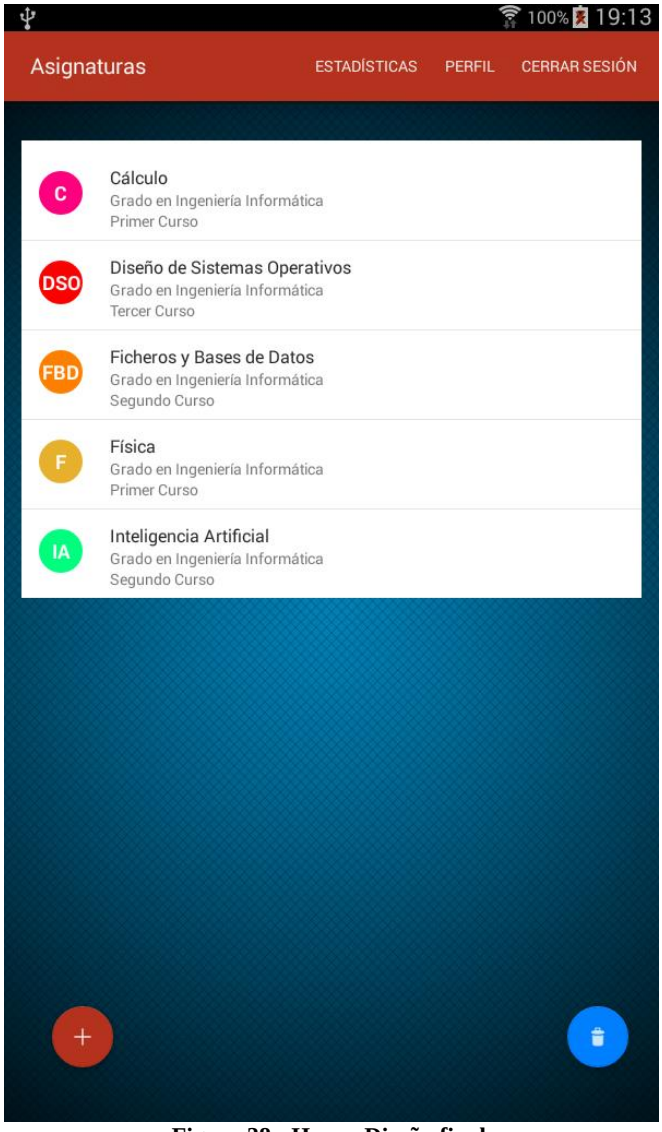

Figura 38 - Home. Diseño final.

4.5.3 Funcionalidad

Para acceder a esta pantalla se tiene que dar uno de estos eventos:

- El usuario inicia sesión y ha utilizado, previamente, la aplicación.
- El usuario utiliza por primera vez la aplicación y accede desde el módulo de inscripción de asignaturas.
- El usuario mantuvo la sesión abierta la última vez que cerró la aplicación. Entonces, al volver a abrir la aplicación, se salta el proceso de login gracias a los datos salvados mediante *SharedPreferences.*

En la *activity* principal (Figura 38) se cargan los datos del usuario almacenados en *SharedPreferences*. En este caso, se utiliza el identificador del usuario para recuperar las asignaturas inscritas, de la base de datos. Se obtiene una colección de asignaturas que se distribuyen en un *ListView*, cada asignatura se coloca en una fila de la lista. Asimismo, se coloca un círculo con el acrónimo de la asignatura, el nombre del grado y el curso acompañando al nombre de la asignatura en cada una de las filas.

El usuario puede seleccionar una de las asignaturas, presentes en la lista, para acceder a su temario.

Como se muestra en la Figura 38, se ha situado dos *Floating Action Button* que permiten al usuario añadir y eliminar asignaturas de la lista, respectivamente.

4.5.4 Flujo de datos

Para la implementación de la sección principal se ha utilizado una arquitectura cliente-servidor. En el lado del servidor se ha utilizado el lenguaje PHP y la base de datos remota MySQL. El lado del cliente (aplicación Android) se comunica con el servidor por medio de HTTP y JSON.

El proceso de comunicación entre el cliente y el servidor se explica en los siguientes puntos. Asimismo, se muestra una representación visual del intercambio de datos, que se genera al acceder en la pantalla principal, en la Figura 39.

1. La aplicación obtiene los identificadores del grado matriculado y del usuario, datos que están almacenados en *SharedPreferences*.
2. La aplicación envía una petición GET al servicio web REST, especificado en el apartado 4.12, con el identificador del grado como parámetro.

3. El servicio web ejecuta el código PHP y envía una consulta a la tabla **Degree** de la base de datos que selecciona el grado asociado al identificador pasado como parámetro.
4. La base de datos devuelve el campo nombre de la fila asociada al grado resultante de la consulta SQL.
5. El servidor PHP envía el dato, nombre de grado, al cliente en formato JSON.
6. La aplicación recibe el nombre del grado y lo almacena en una variable y en *SharedPreferences.*
7. La aplicación envía una petición GET al servicio web REST con el identificador del usuario como parámetro.
8. El servicio web ejecuta el código PHP y envía una consulta a la tabla **User_Subject** de la base de datos que selecciona las asignaturas cuyo usuario matriculado coincida con el identificador pasado como parámetro.
9. La base de datos devuelve los campos: identificador, nombre, acrónimo y curso de las filas asociadas a las asignaturas resultantes de la sentencia SQL.
10. El servidor PHP envía los datos al cliente en formato JSON.
11. La aplicación recibe el identificador, nombre, acrónimo y curso de cada una de las asignaturas y los almacena los *ArrayList.*
12. La aplicación reúne toda la información obtenida de la base de datos y distribuye los datos en una lista.
13. Para cada elemento de la lista, es decir, cada asignatura, se ordena su información, como se puede ver en la Figura 38, de la siguiente manera: acrónimo, nombre de la asignatura, nombre del grado y curso.

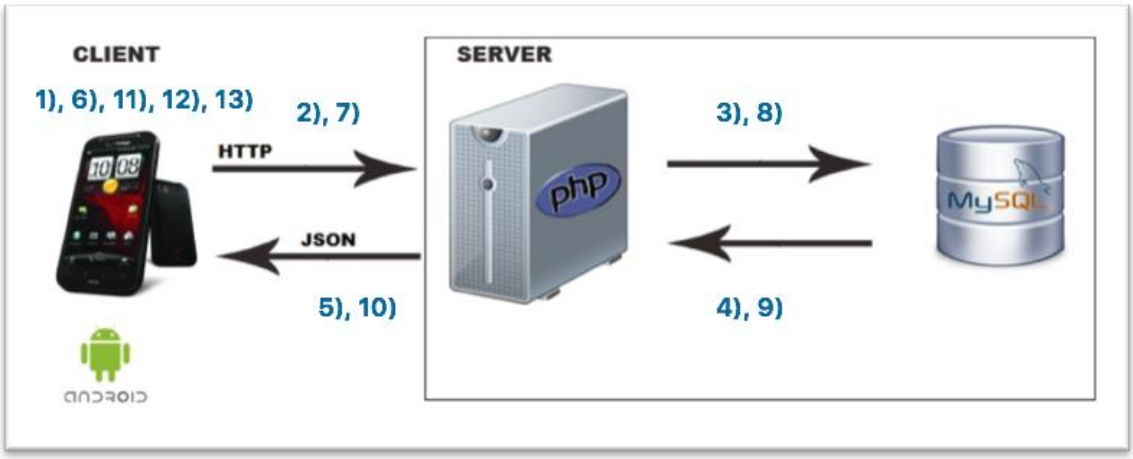

Figura 39 - Home. Flujo de datos.

4.6 Añadir/eliminar asignaturas

4.6.1 Diseño de las pantallas para añadir/eliminar asignaturas

Las pantallas para añadir/eliminar asignaturas, Figura 40, han sido introducidas para añadir la nueva funcionalidad que permita al usuario añadir o eliminar sus asignaturas inscritas. Además, se ha incluido, en ambas pantallas, el menú en el *Action Bar*, en caso de no haber espacio suficiente para distribuir los elementos del menú en el *Action Bar* se establece automáticamente un menú desplegable.

En la pantalla para añadir asignaturas, el diseño final se ha construido con un *ListView* que contiene todas las asignaturas, pertenecientes al grado, que el usuario no ha inscrito. Y un botón que asociará todas las asignaturas, seleccionadas mediante los *checkbox* de la lista, al usuario que las seleccionó y guarda los datos relativos a ambos en la base de datos.

En la pantalla para eliminar asignaturas, el diseño final se ha construido con un *ListView* que contiene únicamente las asignaturas que el usuario ha inscrito. Y un botón que desligará todas las asignaturas, seleccionadas mediante los *checkbox* de la lista, del usuario que las seleccionó.

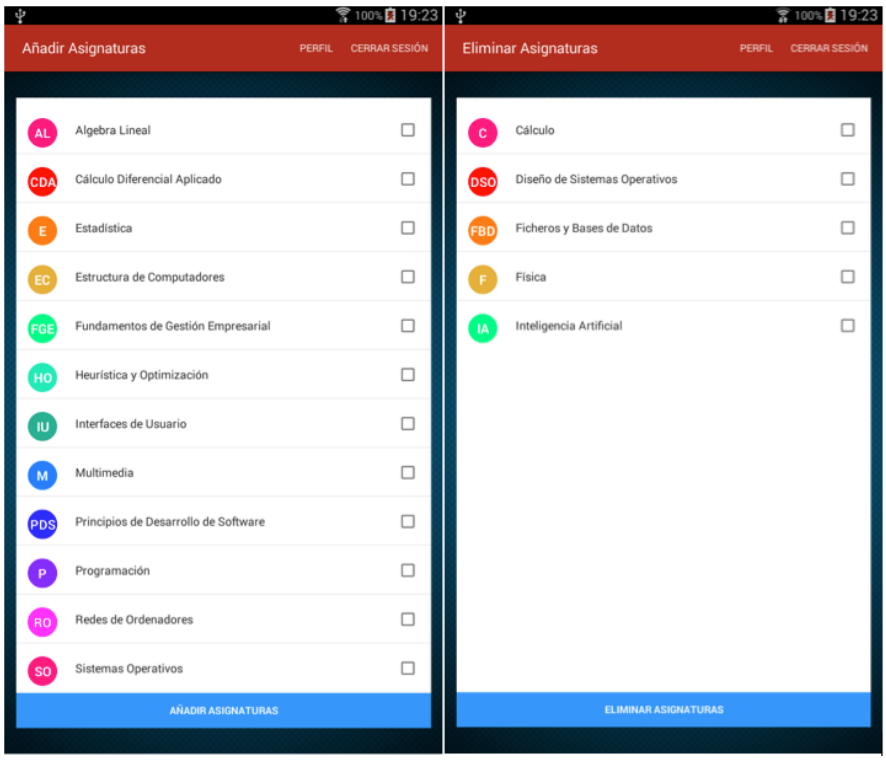

Figura 40 - Añadir/Eliminar asignaturas.

4.6.2 Funcionalidad

En el módulo de añadir/eliminar asignaturas (Figura 40) se cargan los datos del usuario almacenados en *SharedPreferences*.

- En el caso de añadir asignaturas, se utiliza el identificador del grado, matriculado por el usuario, para recuperar las asignaturas no matriculadas por el usuario
- En el caso de eliminar asignaturas, se utiliza el identificador del usuario para recuperar las asignaturas matriculadas de la base de datos.

Se obtiene una colección de asignaturas que se distribuyen en un *ListView*, cada asignatura se coloca en una fila de la lista. Asimismo, se coloca un círculo con el acrónimo de la asignatura y un checkbox acompañando al nombre de la asignatura en cada una de las filas.

En el sub-módulo de añadir asignaturas, el usuario selecciona las asignaturas, que quiera matricular, activando los *checkbox* correspondientes, a continuación, se asocian estas asignaturas al usuario en la base de datos.

En el sub-módulo de eliminar asignaturas, el usuario selecciona las asignaturas, que quiera desmatricular, activando los *checkbox* correspondientes, a continuación, se desligan estas asignaturas del usuario en la base de datos.

4.6.3 Flujo de datos

Para la implementación de la inscripción de asignaturas se ha utilizado una arquitectura cliente-servidor. En el lado del servidor se ha utilizado el lenguaje PHP y la base de datos remota MySQL. El lado del cliente (aplicación Android) se comunica con el servidor por medio de HTTP y JSON.

4.6.3.1 Sub-módulo de añadir asignaturas

El proceso de comunicación entre el cliente y el servidor se explica en los siguientes puntos. Asimismo, se muestra una representación visual del intercambio de datos, que se genera al añadir asignaturas, en la Figura 41.

1. La aplicación obtiene el identificador del grado matriculado por el usuario, dato que está almacenado en *SharedPreferences*.

2. La aplicación envía una petición GET al servicio web RESTful, especificado en el apartado 4.12, con el identificador del grado como parámetro.
3. El servicio web ejecuta el código PHP y envía una consulta a la tabla **Subject** de la base de datos que selecciona todas las asignaturas relativas al grado.
4. La base de datos devuelve los campos de las filas asociadas a las asignaturas resultantes de la sentencia SQL.
5. El servidor PHP envía los datos al cliente en formato JSON.
6. La aplicación recibe el identificador, nombre y acrónimo de cada una de las asignaturas y los almacena en *ArrayList*.
7. Después, la aplicación obtiene el identificador del usuario, dato que está almacenado en *SharedPreferences*.
8. La aplicación envía una petición GET al servicio web RESTful con el identificador del usuario como parámetro.
9. El servicio web ejecuta el código PHP y envía una consulta a la tabla **User_Subject** de la base de datos que selecciona todas las asignaturas matriculadas por dicho usuario.
10. La base de datos devuelve los campos de las filas asociadas a las asignaturas resultantes de la sentencia SQL.
11. El servidor web envía los datos al cliente en formato JSON.
12. La aplicación recibe el identificador, nombre y acrónimo de cada una de las asignaturas y los almacena en *ArrayList*.
13. En este momento, la aplicación posee un conjunto de todas las asignaturas pertenecientes al grado y un conjunto de todas las asignaturas matriculadas por el usuario. Entonces, se realiza una selección de las asignaturas que el usuario no ha matriculado.
14. La aplicación coloca los datos de las asignaturas no matriculadas en una lista.
15. El usuario selecciona las asignaturas que desee matricular y confirma las suscripciones.
16. La aplicación envía tantas peticiones POST como asignaturas haya seleccionado el usuario al servicio web RESTful.
17. A cada petición POST se le añaden los parámetros: identificador de usuario e identificador de asignatura.
18. El servicio web ejecuta el código PHP y envía sentencias a la base datos para matricular al usuario en las asignaturas.

19. En la base de datos MySQL, se crea una nueva entrada en la tabla **User_Subject** con los datos de usuario y de asignatura.

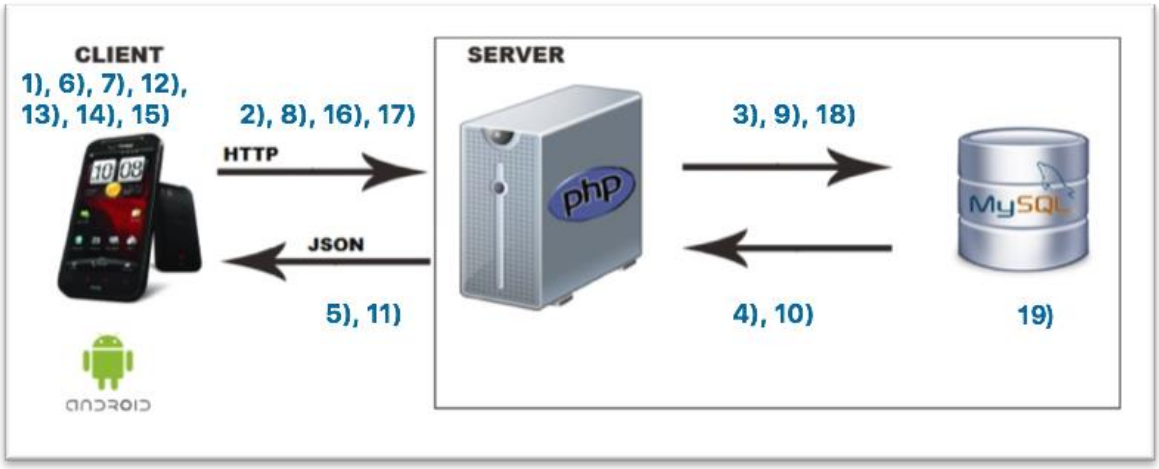

Figura 41 - Añadir asignaturas. Flujo de datos.

4.6.3.2 Sub-módulo de eliminar asignaturas

El proceso de comunicación entre el cliente y el servidor se explica en los siguientes puntos. Asimismo, se muestra una representación visual del intercambio de datos, que se genera al eliminar asignaturas, en la Figura 42.

1. La aplicación obtiene el identificador del usuario, dato que está almacenado en *SharedPreferences*.
2. La aplicación envía una petición GET al servicio web RESTful, especificado en el apartado 4.12, con el identificador del usuario como parámetro.
3. El servicio web ejecuta el código PHP y envía una consulta a la tabla **User_Subject** de la base de datos que selecciona todas las asignaturas matriculadas por dicho usuario.
4. La base de datos devuelve los campos de las filas asociadas a las asignaturas resultantes de la sentencia SQL.
5. El servidor PHP envía los datos al cliente en formato JSON.
6. La aplicación recibe el identificador, nombre y acrónimo de cada una de las asignaturas y los almacena en *ArrayList*.

7. La aplicación reúne toda la información obtenida de la base de datos y distribuye los datos en una lista.
8. Para cada elemento de la lista, es decir, cada asignatura, se ordena su información, como se puede ver en la Figura 40, de la siguiente manera: acrónimo, nombre de la asignatura, checkbox.
9. El usuario selecciona las asignaturas que desee eliminar y confirma la desvinculación.
10. La aplicación envía tantas peticiones POST como asignaturas haya seleccionado el usuario al servicio web RESTful.
11. A cada petición POST se le añaden los parámetros: identificador de usuario e identificador de asignatura.
12. El servicio web ejecuta el código PHP y envía sentencias a la base datos para desvincular al usuario de las asignaturas.
13. En la base de datos MySQL, se borran las filas de la tabla **User_Subject** con los datos de usuario y de asignatura.

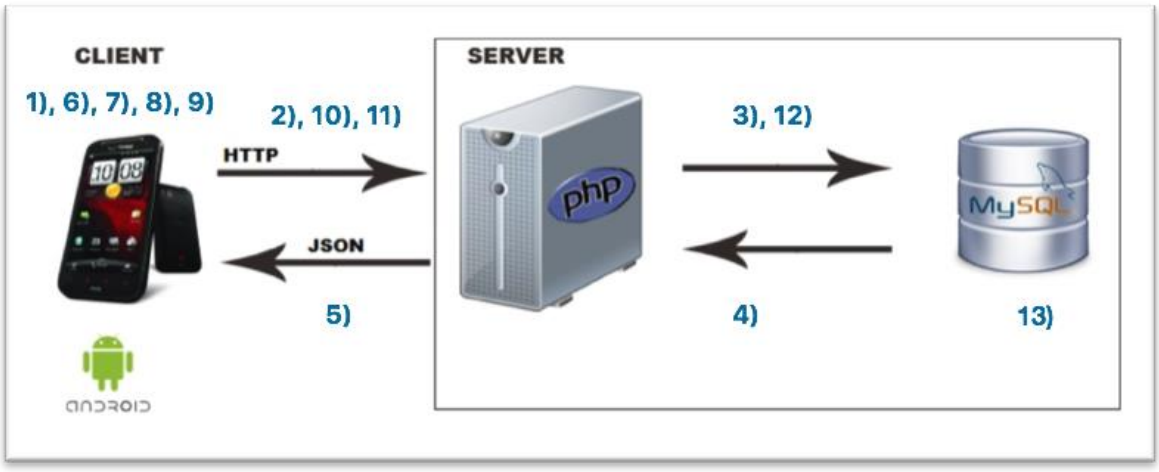

Figura 42 - Eliminar asignaturas. Flujo de datos.

4.7 Temario

4.7.1 Diseño inicial de la pantalla de temario

La Figura 43 muestra el diseño inicial de la pantalla de temario. El propósito de esta pantalla es mostrar el temario de una asignatura en una lista y permitir al usuario acceder a las preguntas. Para ello se ha incluido los siguientes elementos de interfaz:

- Una lista que contiene todos los temas de una asignatura.
    - Cada elemento de la lista se compone de una imagen, el título del tema, el número del tema y la asignatura a la que pertenece.
- Un botón que despliega el menú de opciones.

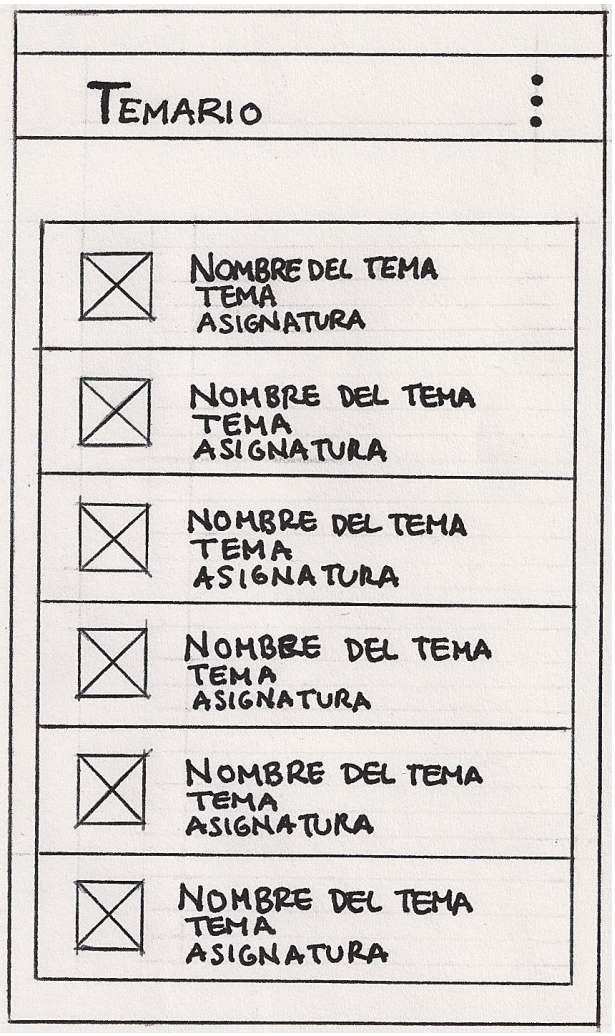

Figura 43 - Temario. Mockup.

4.7.2 Diseño final de la pantalla de temario

La pantalla de temario, Figura 44, se ha modelado siguiendo el diseño presentado en el apartado 4.7.1. El diseño final se ha construido con un *ListView* que contiene todos los temas pertenecientes a una asignatura seleccionada.

Además, se han incluido el menú en el *Action Bar*, en caso de no haber espacio suficiente para distribuir los elementos del menú en el *Action Bar* se establece automáticamente un menú desplegable.

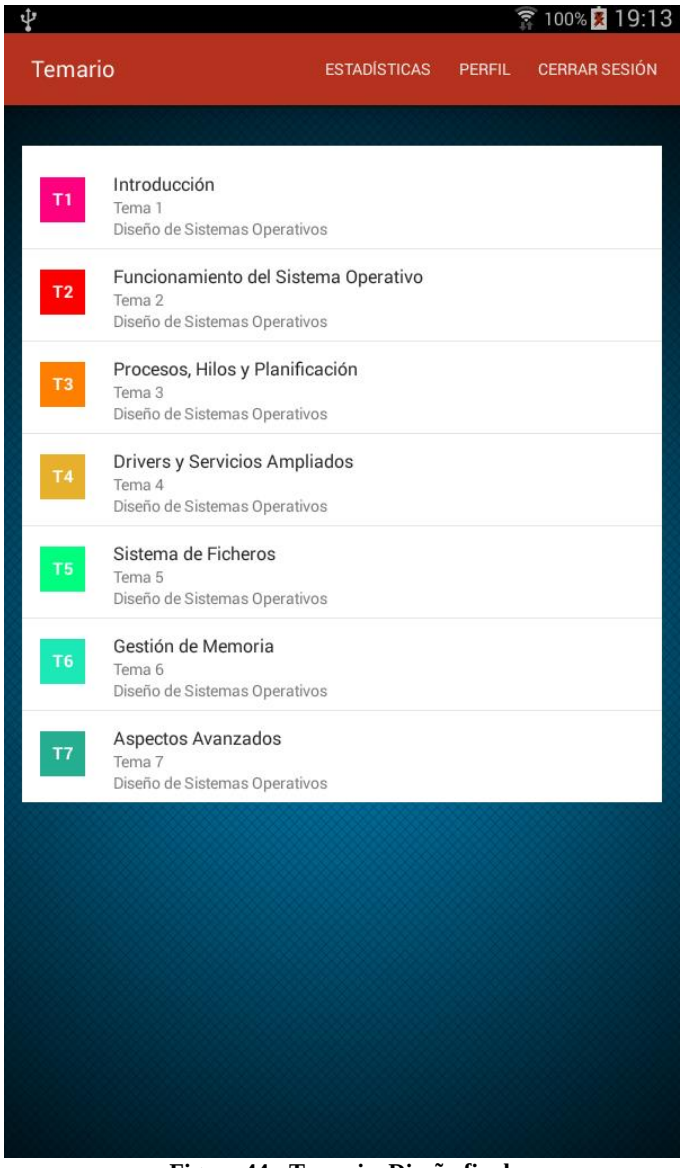

Figura 44 - Temario. Diseño final.

4.7.3 Funcionalidad

En el módulo de temario (Figura 44) se utiliza el identificador de la asignatura para recuperar sus temas correspondientes de la base de datos. Se obtiene una colección de temas que se distribuyen en un *ListView*, cada tema se coloca en una fila de la lista. Asimismo, se coloca una imagen, el número del tema y el nombre de la asignatura acompañando al título del tema en cada una de las filas.

El usuario puede seleccionar uno de los temas, presentes en la lista, para acceder a sus preguntas.

4.7.4 Flujo de datos

Para la implementación de la sección de temario se ha utilizado una arquitectura cliente-servidor. En el lado del servidor se ha utilizado el lenguaje PHP y la base de datos remota MySQL. El lado del cliente (aplicación Android) se comunica con el servidor por medio de HTTP y JSON.

El proceso de comunicación entre el cliente y el servidor se explica en los siguientes puntos. Asimismo, se muestra una representación visual del intercambio de datos, que se genera al acceder a la sección del temario, en la Figura 45.

1. La aplicación obtiene el identificador de la asignatura seleccionada en el módulo principal, dato que está almacenado en *SharedPreferences*.
2. La aplicación envía una petición GET al servicio web RESTful, especificado en el apartado 4.12, con el identificador de la asignatura como parámetro.
3. El servicio web ejecuta el código PHP y envía una consulta a la tabla **Topic** de la base de datos que selecciona los temas cuya asignatura coincida con el identificador pasado como parámetro.
4. La base de datos devuelve los campos: identificador, nombre y número; de las filas asociadas a los temas resultantes de la sentencia SQL.
5. El servidor PHP envía los datos al cliente en formato JSON.
6. La aplicación recibe el identificador, nombre y número de cada uno de los temas y los almacena los *ArrayList*.
7. La aplicación reúne toda la información obtenida de la base de datos y distribuye los datos en una lista.

8. Para cada elemento de la lista, es decir, cada tema, se ordena su información, como se puede ver en la Figura 44, de la siguiente manera: abreviación, título del tema, número del tema y nombre de la asignatura.

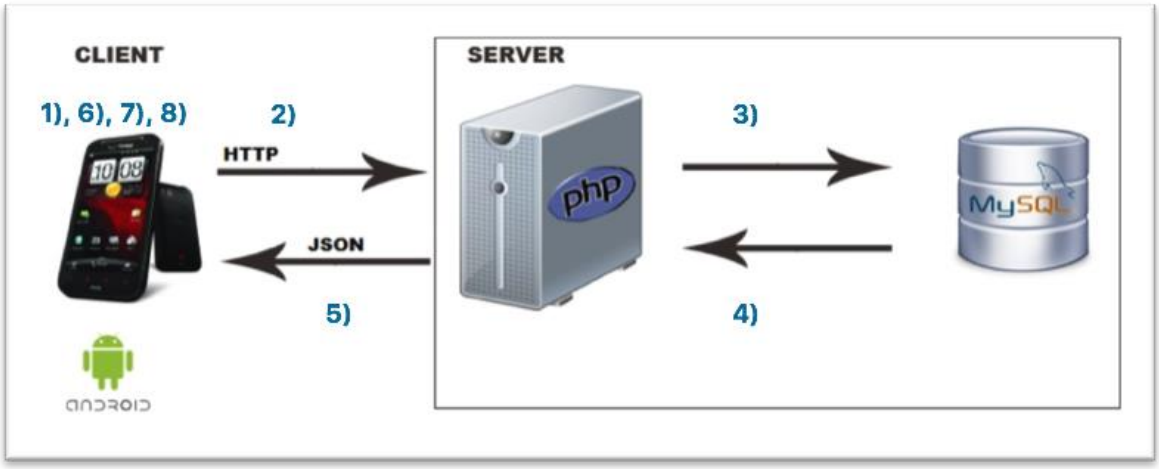

Figura 45 - Temario. Flujo de datos.

4.8 Preguntas

4.8.1 Diseño inicial de la pantalla de preguntas

La Figura 46 muestra el diseño inicial de la pantalla de preguntas. El propósito de esta pantalla es mostrar una pregunta al usuario y sus posibles soluciones. Para ello se ha incluido los siguientes elementos de interfaz:

- Un cuadro de texto que muestra una pregunta.
- Un cuadro de texto que muestra la solución A.
- Un cuadro de texto que muestra la solución B.
- Un cuadro de texto que muestra la solución C.
- Un cuadro de texto que muestra la solución D.
- Un botón que permitirá la respuesta por voz.

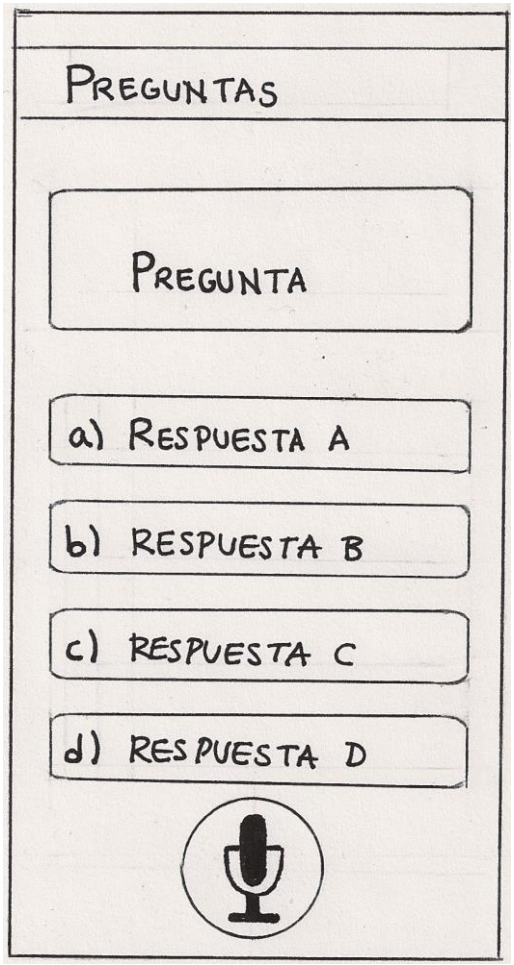

Figura 46 - Preguntas. Mockup.

4.8.2 Diseño final de la pantalla de preguntas

La pantalla de preguntas, Figura 47, se ha modelado siguiendo el diseño presentado en el apartado 4.8.1. El diseño final se ha construido con un *Fragment*, que contiene cada una de las diez preguntas que se realizan en cada ronda de test.

Cada pregunta se compone de cinco *TextView* donde el primero muestra la pregunta y los otros cuatro muestran las respuestas a la pregunta, siendo solamente una de ellas la respuesta correcta; y un botón que activa la función de reconocimiento del habla que permite al usuario contestar a la pregunta mediante voz.

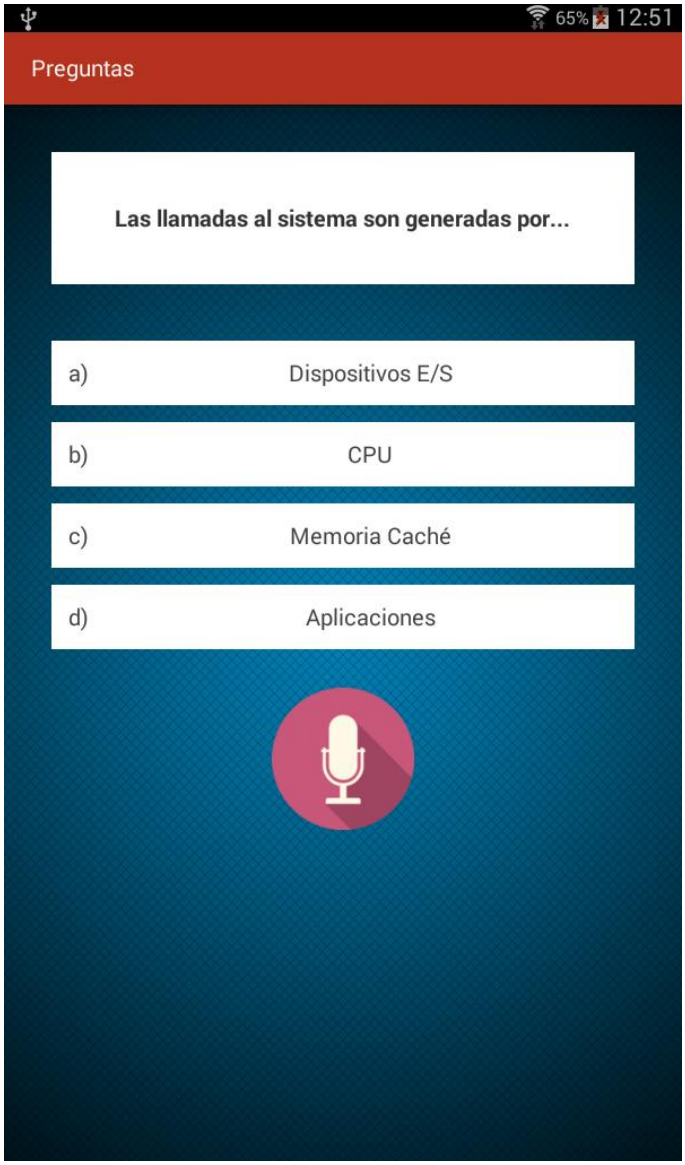

Figura 47 - Preguntas. Diseño final.

4.8.3 Funcionalidad

En el módulo de preguntas (Figura 47) la aplicación obtiene diez preguntas de la base de datos, se muestra una pregunta a la vez. Cuando se contesta la pregunta, se pasa a mostrar la siguiente pregunta hasta un total de diez.

A medida que se van contestando las preguntas, el sistema va guardando la información relativa a las respuestas, es decir, los aciertos o fallos realizados por el usuario.

Se da al usuario la opción de contestar a las preguntas de las siguientes maneras:

- Mediante el uso de la pantalla táctil: El usuario elige una respuesta de entre las cuatro disponibles. Hace *click* en la respuesta, es entonces cuando el sistema determina si la respuesta es correcta o no.
- Mediante el uso del reconocimiento del habla: El usuario hace *click* en el botón con el icono de un micrófono, entonces se ejecuta el asistente de reconocimiento de voz.
    - El usuario puede responder pronunciando la respuesta entera.
    - El usuario puede responder indicando la opción de su respuesta, es decir, el usuario indica que la respuesta es la opción ‘a’, ‘b’, ‘c’ o ‘d’.

  Una vez que el usuario ha proporcionado una respuesta, el sistema determina si la respuesta es correcta, es incorrecta o no coincide con ninguna de las posibles soluciones, en este caso, se permite al usuario repetir la respuesta.

Cuando se resuelve la pregunta, se comunica al usuario si su respuesta es correcta o no, de manera visual, coloreando la respuesta correcta de verde y la incorrecta de color rojo (Figura 48).

- Si la respuesta es correcta, se pinta la opción seleccionada de color verde, que se asocia a algo bueno, por experiencia del usuario.
- Si la respuesta es incorrecta, se pinta la opción seleccionada de color rojo, que se asocia a algo malo, por experiencia. Mientras que la opción correcta se colorea de verde.

Figura 48 - Feedback de respuesta correctas/ incorrectas.

Al mismo tiempo, emerge un *SnackBar*, que informa al usuario si la respuesta es correcta, incorrecta o no coincidente (Figura 49). Además, permite continuar con las siguientes preguntas o volver a intentarlo, en el caso de respuesta no coincidente.

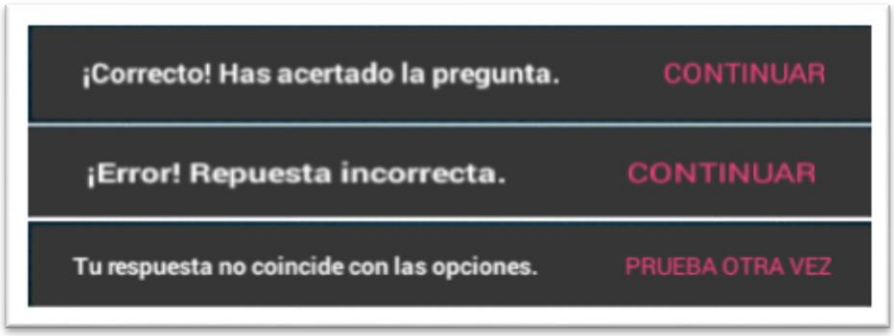

Figura 49 - Snackbar. Feedback.

4.8.4 Flujo de datos

Para la implementación de la sección de preguntas se ha utilizado una arquitectura cliente-servidor. En el lado del servidor se ha utilizado el lenguaje PHP y la base de datos remota MySQL. El lado del cliente (aplicación Android) se comunica con el servidor por medio de HTTP y JSON.

El proceso de comunicación entre el cliente y el servidor se explica en los siguientes puntos. Asimismo, se muestra una representación visual del intercambio de datos, que se genera al responder preguntas, en la Figura 50.

1. La aplicación obtiene el identificador de la asignatura y del tema seleccionado en el módulo de temario, datos que han sido almacenados en *SharedPreferences*.

2. La aplicación envía una petición GET al servicio web REST, explicado en el apartado 4.12, con los identificadores de la asignatura y del tema como parámetros.
3. El servicio web ejecuta el código PHP y envía una consulta a la tabla **Question** de la base de datos que selecciona diez preguntas, de manera aleatoria, cuyas asignaturas y temas coincidan con los identificadores pasados por parámetro.
4. La base de datos devuelve los campos: identificador, pregunta, respuesta1, respuesta2, respuesta3, respuesta4 y solución; de las filas asociadas a las preguntas resultantes de la sentencia SQL.
5. El servidor web envía los datos al cliente en formato JSON.
6. La aplicación recibe el identificador, pregunta, respuesta1, respuesta2, respuesta3, respuesta4 y solución de cada una de las preguntas y los almacena los *ArrayList*.
7. La aplicación crea diez instancias de la clase *Question*, implementada para guardar los datos de cada pregunta.
8. A su vez, las diez instancias se almacenan en un *array* de la clase *Question.* Este *array* se guarda como variable global para que el *Fragment* obtenga los datos de cada una de las preguntas.
9. El *Fragment* posiciona la pregunta y las respuestas en sus respectivos *TextView*.
10. Cuando el usuario responde una pregunta se pasa a la siguiente, hasta cumplir el ciclo de diez preguntas.

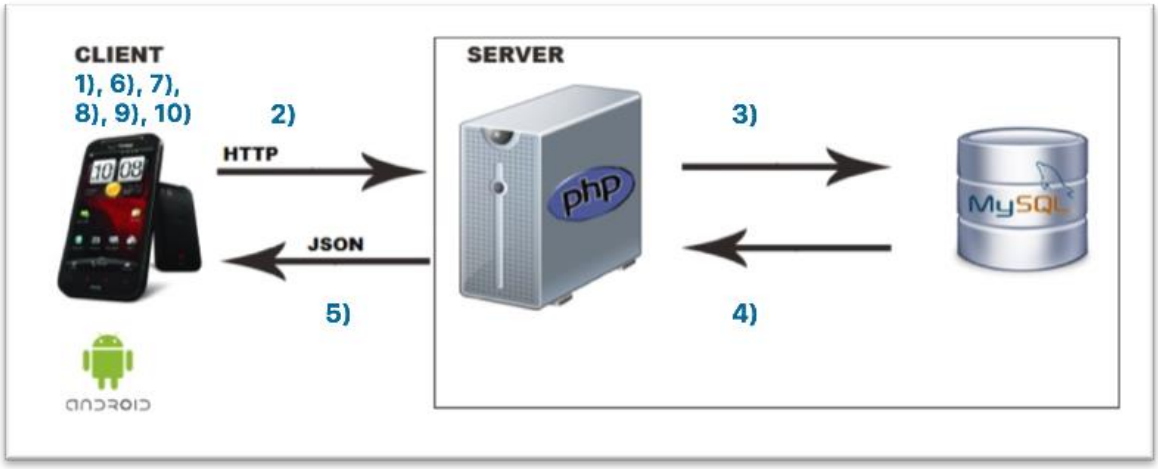

Figura 50 - Preguntas. Flujo de datos.

4.9 Resultados

4.9.1 Diseño inicial de la pantalla de resultados

La Figura 51 muestra el diseño inicial de la pantalla de resultado de test. El propósito de esta pantalla es mostrar los resultados conseguidos al finalizar la ronda de preguntas. Para ello se ha incluido los siguientes elementos de interfaz:

- Un cuadro de texto que muestra el número de aciertos.
- Un cuadro de texto que muestra el número de fallos.
- Un cuadro de texto que muestra la nota obtenida.
- Un botón para regresar a la pantalla principal.

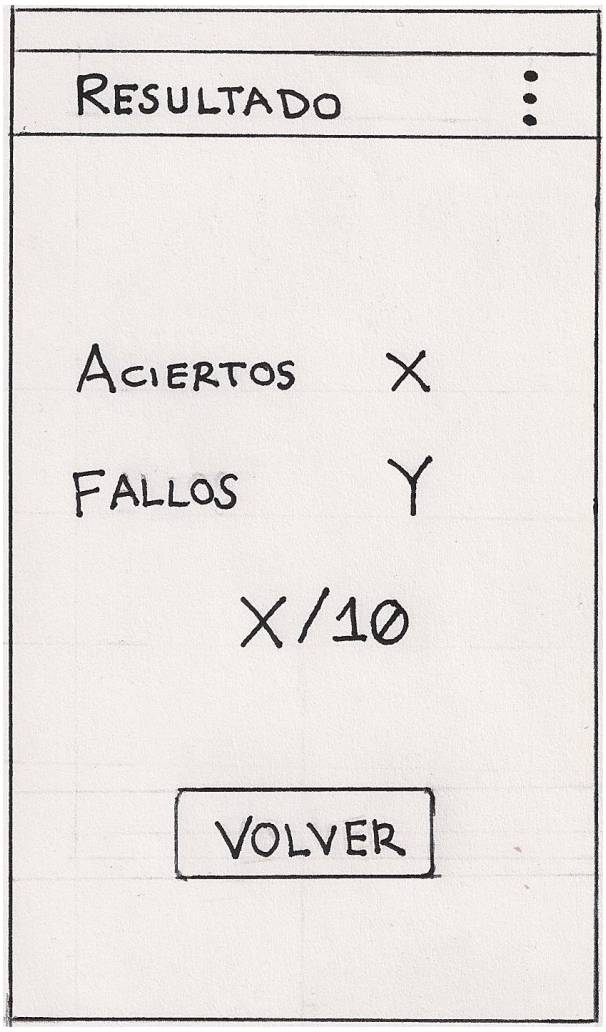

Figura 51 - Resultados. Mockup

4.9.2 Diseño final de la pantalla de resultados

La pantalla de resultado, Figura 52, se ha modelado siguiendo el diseño presentado en el apartado 4.9.1. El diseño final se ha construido con tres *TextView* que muestran los resultados obtenidos, así como la nota obtenida tras realizar la ronda de 10 preguntas; y un botón que permite regresa a la pantalla principal.

Además, se han incluido el menú en el *Action Bar*, en caso de no haber espacio suficiente para distribuir los elementos del menú en el *Action Bar* se establece automáticamente un menú desplegable.

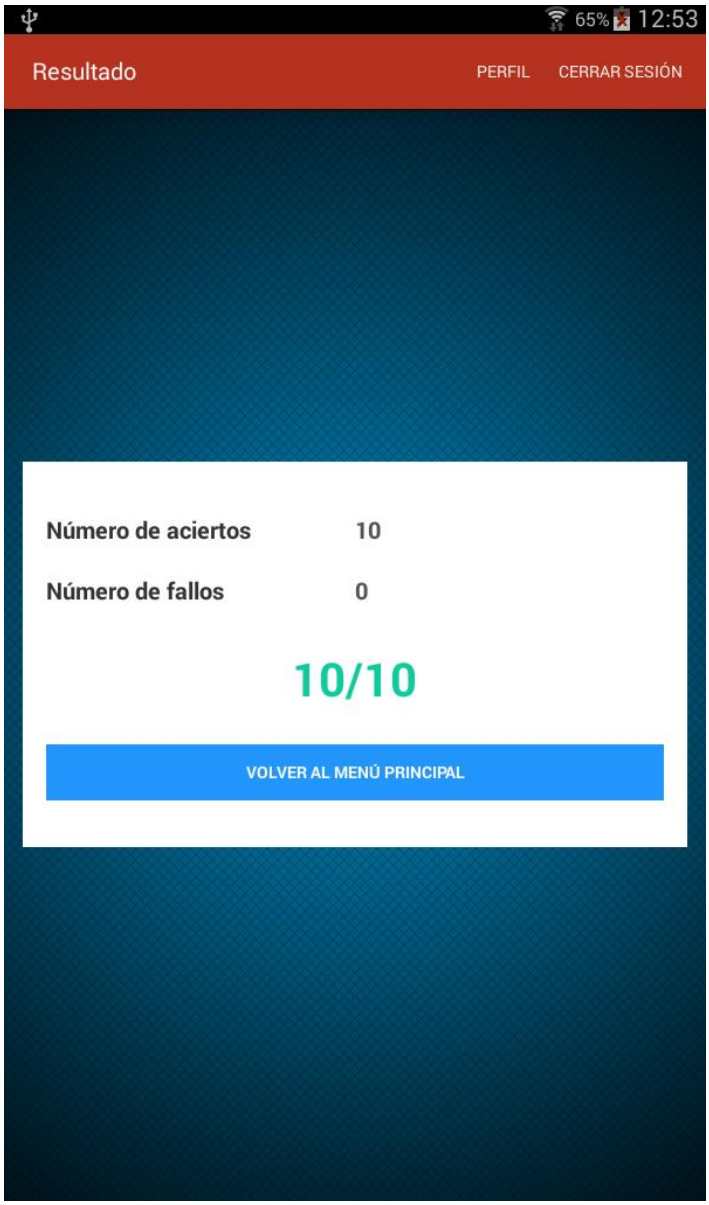

Figura 52 - Resultado. Diseño final.

4.9.3 Funcionalidad

En la sección de resultados (Figura 52) la aplicación obtiene el número de aciertos y fallos que ha cometido el usuario durante la ronda de preguntas y los muestra al usuario.

Este módulo no tiene otra finalidad que presentar al usuario la nota obtenida en el último test realizado. Para que, de esta manera, sepa si debe estudiar el tema con más profundidad.

4.9.4 Flujo de datos

Para la implementación de la sección de resultados se ha utilizado una arquitectura cliente-servidor. En el lado del servidor se ha utilizado el lenguaje PHP y la base de datos remota MySQL. El lado del cliente (aplicación Android) se comunica con el servidor por medio de HTTP y JSON.

El proceso de comunicación entre el cliente y el servidor se explica en los siguientes puntos. Asimismo, se muestra una representación visual del intercambio de datos, que se genera al finalizar una ronda de preguntas, en la Figura 53.

1. La aplicación obtiene el identificador del usuario, de la asignatura y del tema seleccionado en el módulo de temario, datos que han sido almacenados en *SharedPreferences*.
2. La aplicación envía una petición GET al servicio web RESTful, explicado en el apartado 4.12, con los identificadores de usuario, de asignatura y de tema como parámetros.
3. El servicio web ejecuta el código PHP y envía una consulta a la tabla **User_Topic** de la base de datos que selecciona el número de aciertos y el número de test realizados en ese tema por el usuario en total; cuyos usuario, asignatura y tema coincidan con los identificadores pasados por parámetro.
4. La base de datos devuelve los campos: identificador, número de aciertos y número de tests; de las filas resultantes de la sentencia SQL.
5. El servidor web envía los datos al cliente en formato JSON.
6. La aplicación recibe los datos que vinculan al usuario con el tema.
7. Se suma los resultados obtenidos en el último test con los resultados totales. Entonces, se suma el número de aciertos y el número de aciertos totales. También, se suma una unidad al número de tests realizados.

8. La aplicación envía una petición POST al servicio web RESTful con los datos test sumados.
9. El servicio web ejecuta el código PHP y envía una sentencia a la tabla **User_Topic** de la base de datos para actualizar los resultados.
10. En la base de datos MySQL, se actualiza el número de aciertos y el número de test realizados, en total, por el usuario en el tema actual.

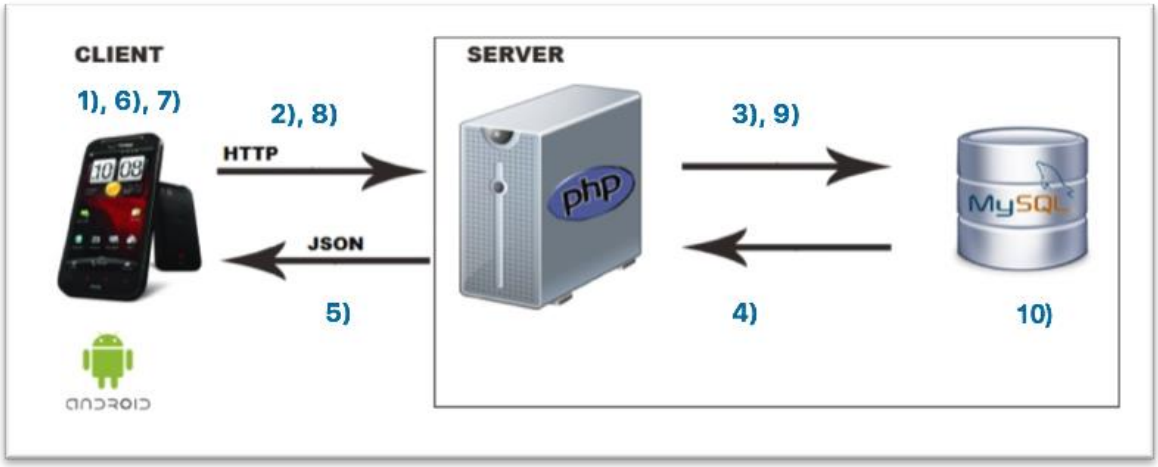

Figura 53 - Resultado. Flujo de datos.

4.10 Estadísticas de asignaturas/temas

4.10.1 Pantalla de estadísticas de asignaturas/temas

Las pantallas para estadísticas de asignaturas y temas, Figura 54, han sido introducidas para añadir una nueva funcionalidad que permita al usuario visualizar los registros logrados tras la realización de los test. Además, se ha incluido, en ambas pantallas, el menú en el *Action Bar*.

En la pantalla de estadísticas de asignaturas, el diseño final se ha construido con un *ListView* que contiene todas las asignaturas inscritas por el usuario, el número de test realizados para cada asignatura y la nota media obtenida para cada una de ellas.

En la pantalla de estadísticas de temas, el diseño final se ha construido con un *ListView* que contiene los temas desglosados, de una asignatura específica, el número de test realizados para cada tema y la nota media obtenida para cada una de ellos.

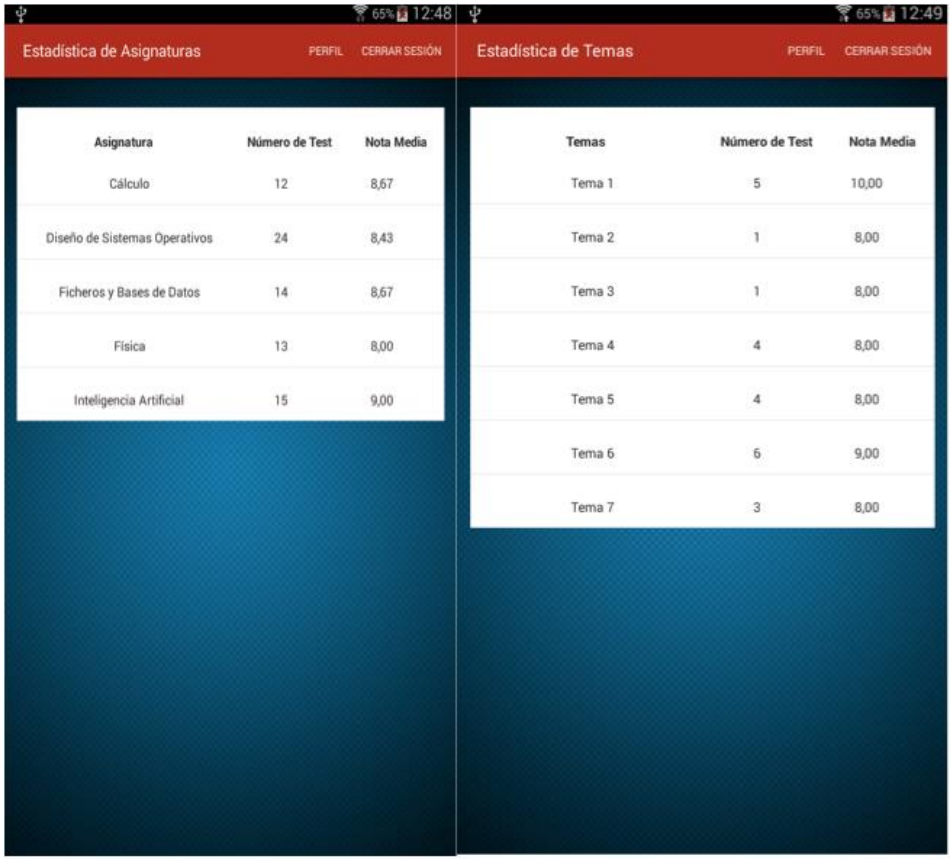

Figura 54 - Estadística de asignaturas/ temas. Diseño final.

4.10.2 Funcionalidad

En el módulo de estadísticas de asignaturas/temas (Figura 54) se cargan los resultados totales obtenidos por el usuario tras realizar tests.

En el sub-módulo de estadísticas de temas, se muestran los resultados desglosados por temas. Se obtienen los datos registrados en la tabla **User_Topic**, concretamente, el número de aciertos y número de test realizados en cada tema.

- Se muestra el número de test realizados, de esta manera, el usuario puede hacerse una idea del progreso que ha seguido en los distintos temas.
- Se calcula la nota media obtenida en cada tema, para ello se utiliza la fórmula presentada en la Figura 55. Así, el usuario puede saber qué temas lleva mejor preparados y cuáles necesita estudiar.

$$Nota\ media = \frac{Número\ de\ aciertos}{Número\ de\ tests\ realizados}$$

Figura 55 - Fórmula para calcular la nota media.

En el sub-módulo de estadísticas de asignaturas, se muestran los resultados desglosados por asignaturas. Se obtienen los datos registrados en la tabla **User_Topic**, concretamente, el número de aciertos y número de test realizados en cada tema.

- Se muestra el número de test realizados en todos los temas de la asignatura. Para ello, se realiza la suma del número de test de cada uno de los temas.
- Se calcula la nota media obtenida en cada asignatura, para ello se suma las notas medias de todos los temas utilizando la fórmula presentada en la Figura 55. Así, el usuario puede saber qué asignaturas lleva mejor preparados y cuáles necesita estudiar.

Se obtiene una colección de asignaturas/temas que se distribuyen en un *ListView*, cada asignatura/tema se coloca en una fila de la lista. Asimismo, se sitúa el número de tests realizados y la nota media correspondiente para cada asignatura/tema.

4.10.3 Flujo de datos

Para la implementación del módulo de estadísticas de asignaturas/temas se ha utilizado una arquitectura cliente-servidor. En el lado del servidor se ha utilizado el lenguaje PHP y la base de datos remota MySQL. El lado del cliente (aplicación Android) se comunica con el servidor por medio de HTTP y JSON.

4.10.3.1 Sub-módulo de estadísticas de temas

El proceso de comunicación entre el cliente y el servidor se explica en los siguientes puntos. Asimismo, se muestra una representación visual del intercambio de datos, que se genera al acceder a la sección de estadísticas de temas, en la Figura 56.

1. La aplicación obtiene el identificador del usuario y de la asignatura actual, datos que están almacenados en *SharedPreferences*.
2. La aplicación envía una petición GET al servicio web RESTful, especificado en el apartado 4.12, con el identificador de usuario y de asignatura como parámetro.
3. El servicio web ejecuta el código PHP y envía una consulta a la tabla **User_Topic** de la base de datos que selecciona todos los temas relativos al usuario y a la asignatura enviados como parámetros.
4. La base de datos devuelve los campos número de aciertos y número de tests realizados; de las filas asociadas a los temas resultantes de la sentencia SQL.
5. El servidor PHP envía los datos al cliente en formato JSON.
6. La aplicación recibe el número de aciertos y número de tests realizados de cada uno de los temas.
7. La aplicación reúne toda la información obtenida de la base de datos y distribuye los datos en una lista.
8. Para cada elemento de la lista, es decir, cada tema, se ordena su información, como se puede ver en la Figura 54, de la siguiente manera: número del tema, número de tests realizados y nota media.

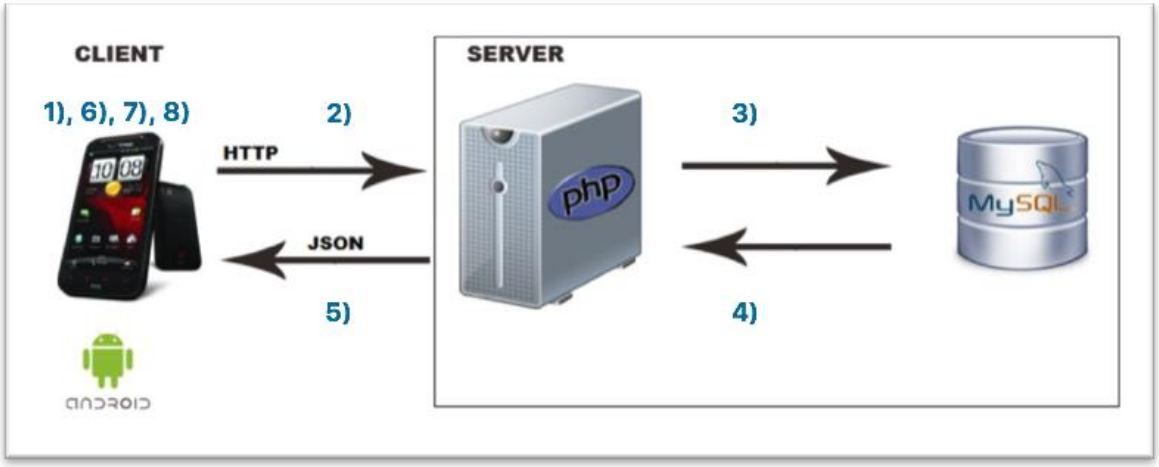

Figura 56 - Estadística de asignaturas. Flujo de datos.

4.10.3.2 Sub-módulo de estadísticas de asignaturas

El proceso de comunicación entre el cliente y el servidor se explica en los siguientes puntos. Asimismo, se muestra una representación visual del intercambio de datos, que se genera al acceder a la sección de estadísticas de asignaturas, en la Figura 57.

1. La aplicación obtiene el identificador del usuario, datos que está almacenado en *SharedPreferences*.
2. La aplicación envía una petición GET al servicio web RESTful, especificado en el apartado 4.12, con el identificador de usuario como parámetro.
3. El servicio web ejecuta el código PHP y envía una consulta a la tabla **User_Subject** de la base de datos que selecciona todas las asignaturas relativos al usuario enviado como parámetro.
4. La base de datos devuelve el identificador y el nombre; de las filas asociadas a las asignaturas resultantes de la sentencia SQL.
5. El servidor PHP envía los datos al cliente en formato JSON.
6. La aplicación recibe el identificador y el nombre de cada una de las asignaturas.
7. La aplicación envía tantas peticiones GET como asignaturas matriculadas al servicio web RESTful con el identificador de usuario y el identificador de asignatura como parámetro.

8. El servicio web ejecuta el código PHP y envía una consulta a la tabla **User_Topic** de la base de datos que selecciona todos los temas relativos al usuario y a la asignatura enviados como parámetros.
9. La base de datos devuelve los campos número de aciertos y número de tests realizados; de las filas asociadas a los temas resultantes de la sentencia SQL.
10. El servidor web envía los datos al cliente en formato JSON.
11. La aplicación reúne toda la información obtenida de la base de datos, calcula los tests realizados en total, las notas medias y distribuye los datos en una lista.
12. Para cada elemento de la lista, es decir, cada asignatura, se ordena su información, como se puede ver en la Figura 54, de la siguiente manera: nombre de la asignatura, número de tests realizados y nota media.

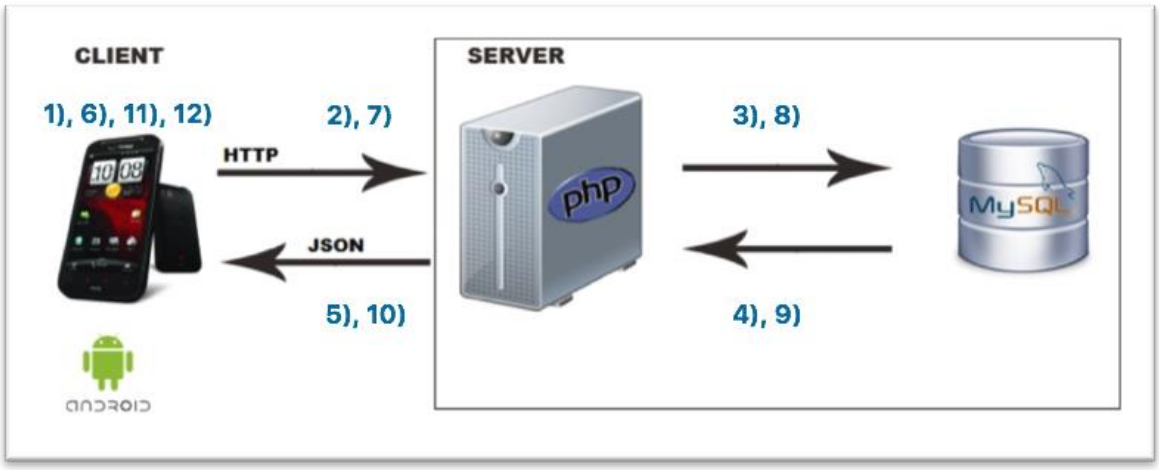

Figura 57 - Estadísticas de asignaturas. Flujo de datos.

4.11 Perfil

4.11.1 Diseño inicial de la pantalla de perfil

La Figura 58 muestra el diseño inicial de la pantalla de perfil. El propósito de esta pantalla es mostrar los datos del usuario y la posibilidad de editarlos. Para ello se ha incluido los siguientes elementos de interfaz:

- Un cuadro de texto que muestra la dirección de correo electrónico.
- Un cuadro de texto que muestra el nombre de usuario.
- Un cuadro de texto que contiene la contraseña.
- Un cuadro de texto que contiene el grado.
- Tres botones para editar nombre de usuario, contraseña y grado.
- Un botón para guardar los cambios.

Figura 58 - Perfil. Mockup.

4.11.2 Diseño final de la pantalla de perfil

La pantalla de perfil, Figura 59, se ha modelado siguiendo el diseño presentado en el apartado 4.11.1. El diseño final se ha construido tres *TextView* que muestran el email, la contraseña y el grado; dos *ImageButton* que permite editar la contraseña y el grado, respectivamente; un *EditText* para editar el nombre de usuario y un botón para guardar los cambios.

Además, se han incluido el menú en el *Action Bar*, en caso de no haber espacio suficiente para distribuir los elementos del menú en el *Action Bar* se establece automáticamente un menú desplegable.

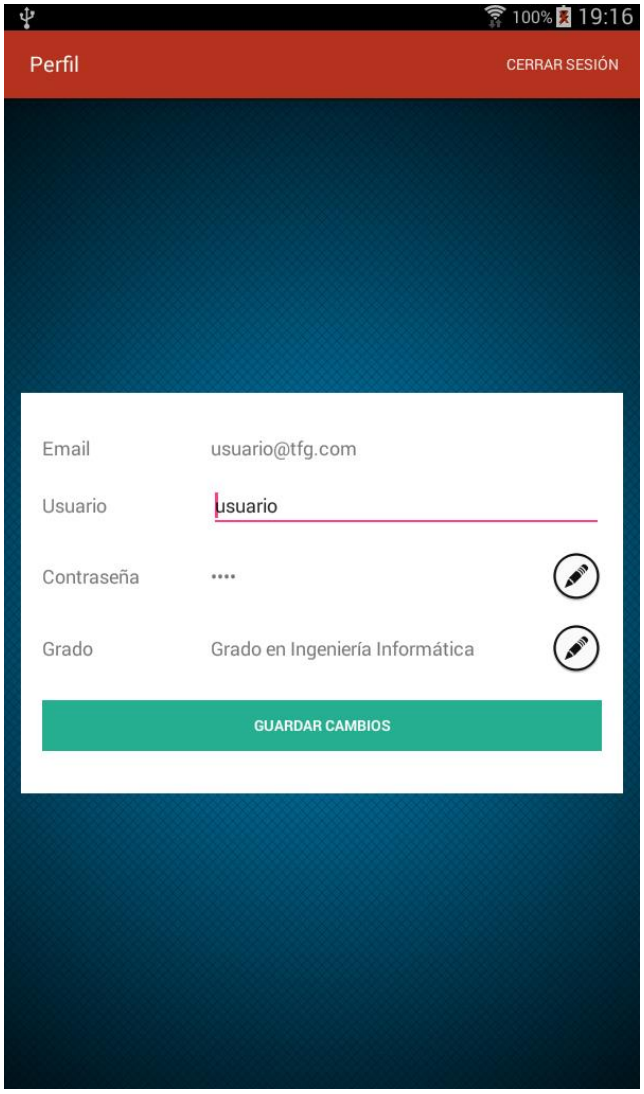

Figura 59 - Perfil. Diseño final.

4.11.3 Funcionalidad

En la sección que corresponde al perfil (Figura 59) se da la posibilidad al usuario de consultar los datos relacionados a sus cuentas. Asimismo, los datos de usuario pueden ser modificados si se desea:

- Nombre de usuario: El usuario puede modificar su nombre con el que inicia sesión.
- Contraseña: El usuario puede modificar la clave con la que inicia sesión.
- Grado académico: El usuario puede cambiar el grado de estudios si desea estudiar materias o aignaturas pertenecientes a otro campo.
- Correo electrónico: El correo electrónico no puede modificarse, puesto que es el identificador de un usuario y se utiliza para recuperar la contraseña.

4.11.4 Flujo de datos

Para la implementación de la sección de perfil de usuario se ha utilizado una arquitectura cliente-servidor. En el lado del servidor se ha utilizado el lenguaje PHP y la base de datos remota MySQL. El lado del cliente (aplicación Android) se comunica con el servidor por medio de HTTP y JSON.

El proceso de comunicación entre el cliente y el servidor se explica en los siguientes puntos. Asimismo, se muestra una representación visual del intercambio de datos, que se genera al acceder a la sección de perfil de usuario, en la Figura 60.

1. La aplicación obtiene el identificador del usuario, que ha sido almacenado en *SharedPreferences*.
2. La aplicación envía una petición GET al servicio web RESTful, explicado en el apartado 4.12, con el identificador de usuario como parámetro.
3. El servicio web ejecuta el código PHP y envía una consulta a la tabla **User** de la base de datos que selecciona los datos de usuario que corresponden al identificador pasado por parámetro.
4. La base de datos devuelve los campos: identificador, nombre de usuario, contraseña, email y grado de la fila resultante de la consulta SQL.
5. El servidor web envía los datos al cliente en formato JSON.
6. La aplicación recibe los datos del usuario.
7. La aplicación coloca la información con el orden visto en la Figura 59. Asimismo, se colocan botones al lado de cada uno de los datos editables.

8. Siempre que un usuario pulse un botón de edición, la aplicación permite editar el texto que muestra el dato correspondiente al botón. Una vez se ha editado el texto, el usuario debe confirmar la modificación de dicho dato.
9. Entonces, la aplicación envía una petición POST al servicio web RESTful con el dato modificado.
10. El servicio web ejecuta el código PHP y envía una sentencia a la tabla **User** de la base de datos para actualizar la información de usuario.
11. En la base de datos MySQL, se actualiza la información modificada.
12. Cuando el usuario vuelva a consultar su perfil, podrá verificar que los cambios se han efectuado.

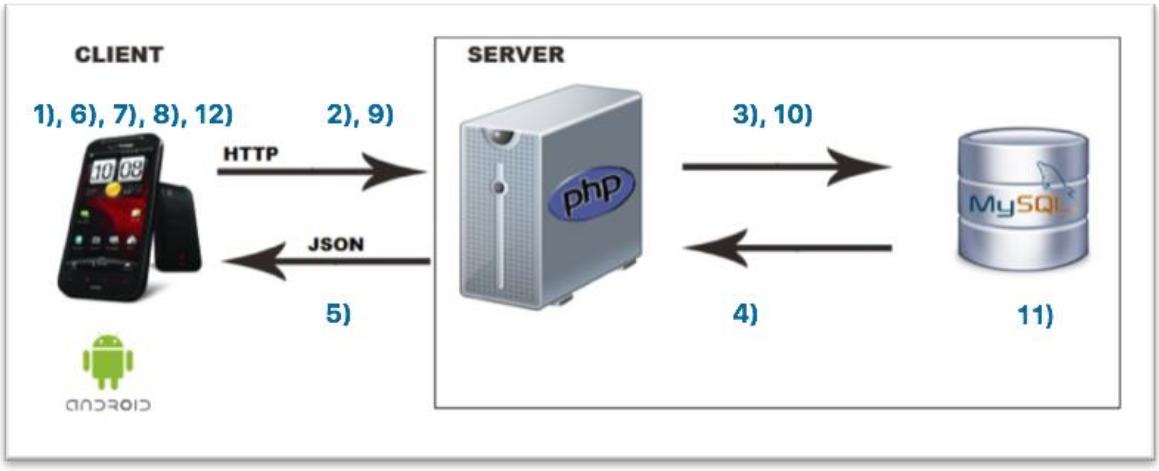

Figura 60 - Perfil. Flujo de datos.

4.12 Servicio web

Un servicio web es una tecnología que proporciona una API cuyos métodos permiten realizar diferentes acciones entre la aplicación cliente y el servidor. El servicio web proporciona interoperabilidad entre el cliente y el servidor vía internet.

El servicio web RESTful permite que el cliente realice peticiones al servidor mediante HTTP, que proporciona los métodos GET, POST, PUT, DELETE, etc. Asimismo, se generan respuestas a cada una de las solicitudes en formato JSON, que informan al usuario acerca del resultado de la operación [43] (ver Figura 61 [44]).

El servicio web está implementado en el lenguaje PHP haciendo uso de diferentes ficheros. Estos ficheros están alojados en un servidor proporcionado por el servicio

de hosting, Hostinger (ver apartado 3.4.2). Se ha subido los ficheros al servidor utilizando Fillezilla (ver apartado 3.4.1).

El servicio web se compone de una serie de ficheros PHP, éstos ficheros proporcionan métodos para realizar operaciones en la base de datos. Los ficheros que componen el servicio web son:

- Login.php: Se encarga de autenticar al usuario y realizar la conexión con la base de datos.
- User.php: Contiene las funciones que se utilizan para consultar los datos relativos a un usuario, para insertar un nuevo usuario y para modificar un usuario ya existente.
- Degree.php: Contiene las funciones que se utilizan para consultar la información relativa a los grados, en la base de datos.
- Subject.php: Contiene las funciones que se utilizan para consultar la información acerca de las asignaturas, en la base de datos.
- Topic.php: Contiene las funciones que se utilizan para consultar la información sobre los temas y las funciones que se emplean para almacenar las puntuaciones obtenidas por los usuarios.
- Question.php: Contiene las funciones que se utilizan para recuperar la información acerca de las preguntas, en la base de datos.

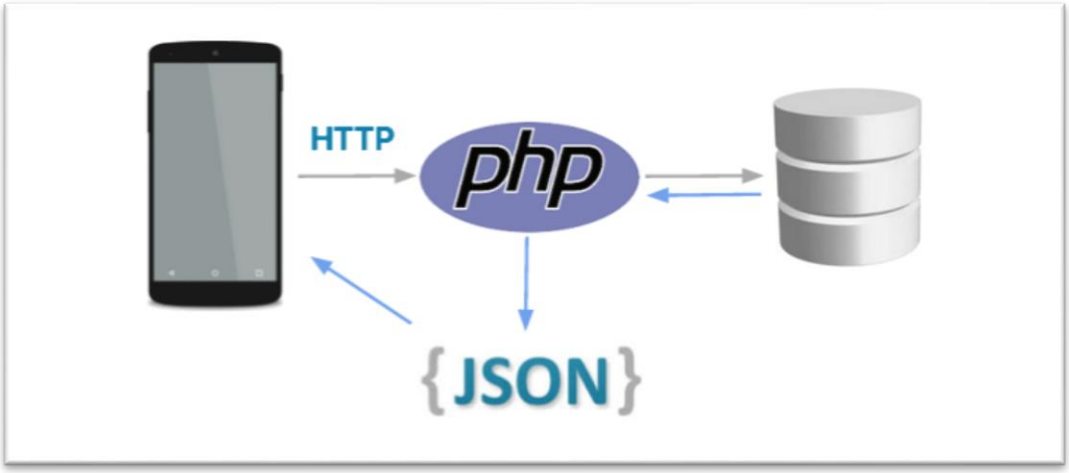

Figura 61 - Servicio web.

Capítulo 5

5. Evaluación de la aplicación

En este capítulo se detallan las pruebas de sistema necesarias para certificar que el sistema desarrollado cumple con los objetivos planteados y que su funcionamiento es adecuado.

Las pruebas se especifican siguiendo la estructura definida en la Tabla 28:

Identificador de la prueba	
Descripción	Breve descripción de la prueba.
Resultado correcto	Resultado esperado de la realización de la prueba.
Resultado obtenido	Resultado obtenido al realizar la prueba.

Tabla 28. Formato de las pruebas.

- El campo de identificador de pruebas se utiliza para identificar a cada una de las pruebas realizadas. Este identificador sigue el formato PX-YY. Donde X expresa el tipo de prueba, es decir, indica si es una prueba unitaria o de integración mediante 'U', e 'I', respectivamente. Mientras que YY indica el número de la prueba, que comienza a partir del 00.
- En el campo de descripción se realiza una explicación breve y concisa de la prueba.
- El campo de resultado correcto contiene el resultado que se espera del sistema.
- El campo de resultado obtenido contiene el resultado logrado como consecuencia de realizar la prueba.

5.1 Pruebas unitarias

Las pruebas unitarias examinan y verifican el correcto funcionamiento de un módulo software. Las pruebas se realizan para comprobar que los módulos que componen la aplicación funcionan adecuadamente.

PU-01	
Descripción	Se prueba que no se pueda registrar un nombre de usuario que ya esté en uso.
Resultado correcto	Se espera un mensaje de error que indique que el nombre usuario no está disponible.
Resultado obtenido	Se recibe un mensaje de error que indica que el nombre usuario no está disponible.

Tabla 29. Prueba unitaria 1.

PU-02	
Descripción	Se prueba que durante el inicio de sesión el nombre de usuario y contraseña introducidos se correspondan a los datos registrados.
Resultado correcto	Se espera un mensaje de error que indique que el nombre usuario y la contraseña no coinciden.
Resultado obtenido	Se recibe un mensaje de error que indica que el nombre usuario y la contraseña no coinciden.

Tabla 30. Prueba unitaria 2.

PU-03	
Descripción	Se prueba que en el módulo de añadir asignaturas (ver Figura 40) aparezcan, únicamente, las asignaturas que el usuario no ha inscrito.
Resultado correcto	Se espera una lista de todas las asignaturas pertenecientes al grado matriculado menos las asignaturas inscritas, es decir, las asignaturas no inscritas.
Resultado obtenido	Se recibe una lista con las asignaturas no inscritas.

Tabla 31. Prueba unitaria 3.

PU-04	
Descripción	Se prueba que en el módulo de eliminar asignaturas (ver Figura 40) aparezcan, únicamente, las asignaturas que el usuario ha inscrito.
Resultado correcto	Se espera una lista de las asignaturas inscritas por el usuario.
Resultado obtenido	Se recibe una lista de las asignaturas inscritas por el usuario.

Tabla 32. Prueba unitaria 4.

PU-05	
Descripción	Se prueba que en el módulo de estadísticas de asignaturas (ver Figura 54) aparecen los resultados correspondientes a cada asignatura.
Resultado correcto	Se espera un listado con las asignaturas y sus correspondientes notas medias y número de test realizados.
Resultado obtenido	Se recibe una lista con las asignaturas y sus correspondientes notas medias y número de test realizados.

Tabla 33. Prueba unitaria 5.

PU-06	
Descripción	Se prueba que en el módulo de estadísticas de temas (ver Figura 54) aparecen los resultados correspondientes a cada tema.
Resultado correcto	Se espera un listado con las notas medias y número de test realizados en una asignatura desglosado por temas.
Resultado obtenido	Se recibe una lista con las notas medias y número de test realizados en una asignatura desglosado por temas.

Tabla 34. Prueba unitaria 6.

PU-07	
Descripción	Se prueba a contestar una pregunta, mediante voz, con una respuesta que no coincide con ninguna de las cuatro posibilidades en el módulo de preguntas (ver Figura 47).
Resultado correcto	Se espera un mensaje del sistema que informe al usuario que su respuesta no coincide con las opciones.
Resultado obtenido	Se recibe un mensaje del sistema que informe al usuario que su respuesta no coincide con las opciones y que permite dar una nueva respuesta que sí coincida con una de las opciones (ver Figura 49).

Tabla 35. Prueba unitaria 7.

PU-08	
Descripción	Se prueba en el módulo de preguntas que el sistema acepte responder mediante voz a las preguntas contestando ‘a’, ‘b’, ‘c’ o ‘d’.
Resultado correcto	Se espera que la aplicación tome la respuesta como correcta o incorrecta según la opción elegida.
Resultado obtenido	La aplicación acepta las respuestas.

Tabla 36. Prueba unitaria 8.

5.2 Pruebas de integración

Las pruebas de integración se aplican para examinar y verificar el correcto funcionamiento de grupos de módulos de manera conjunta. Estas pruebas integran diferentes pruebas unitarias.

PI-01	
Descripción	Se prueba que el inicio de sesión funciona con los datos registrados por el usuario. - El usuario se registra en la aplicación. - El usuario inicia sesión en la aplicación.
Resultado correcto	El resultado esperado es que el sistema permita al usuario a acceder a la aplicación.
Resultado obtenido	La aplicación permite a un usuario iniciar sesión en la aplicación correctamente.

Tabla 37. Prueba de integración 1.

PI-02	
Descripción	Se prueba que el sistema inscribe las asignaturas que elige el usuario.
Resultado correcto	La aplicación debe mostrar las asignaturas inscritas en la pantalla principal.
Resultado obtenido	La aplicación muestra las asignaturas inscritas en la pantalla principal correctamente.

Tabla 38. Prueba de integración 2.

PI-03	
Descripción	Se prueba que el sistema es capaz de añadir nuevas asignaturas a la previamente inscritas.
Resultado correcto	El sistema debe mostrar las nuevas asignaturas en la lista de asignaturas de la pantalla principal.
Resultado obtenido	La aplicación muestra las nuevas asignaturas en la pantalla principal.

Tabla 39. Prueba de integración 3.

PI-04	
Descripción	Se prueba que el sistema es capaz de eliminar asignaturas de entre las previamente inscritas
Resultado correcto	Las asignaturas desapuntadas no deben aparecer en la lista de asignaturas inscritas.
Resultado obtenido	La aplicación muestra la lista de asignaturas inscritas, sin las asignaturas desapuntadas.

Tabla 40. Prueba de integración 4.

PI-05	
Descripción	Se prueba que se realiza un recuento de las preguntas acertadas y falladas a lo largo de un test.
Resultado correcto	La aplicación tiene que mostrar el resultado final de la realización de un test.
Resultado obtenido	Se muestra el resultado y la nota obtenida después de haber realizado un test.

Tabla 41. Prueba de integración 5.

PI-06	
Descripción	Se comprueba que los cambios realizados en nombre de usuario o contraseña se hacen efectivos.
Resultado correcto	Se debe poder iniciar sesión con el nombre de usuario o contraseña nuevos.
Resultado obtenido	Se inicia sesión correctamente tanto con un nombre de usuario nuevo, como con una contraseña nueva.

Tabla 42. Prueba de integración 6.

PI-07	
Descripción	Se prueba que el sistema permite al usuario cambiar el grado matriculado.
Resultado correcto	Se espera que el cambio de grado habilite un nuevo conjunto de asignaturas disponibles para su inscripción y desasocie las asignaturas del grado antiguo.
Resultado obtenido	El sistema cambia el grado, permite la inscripción de las asignaturas del grado nuevo y quita las anteriores asignaturas inscritas.

Tabla 43. Prueba de integración 7.

5.3 Resultado de las pruebas

El sistema cumple con todas las pruebas que se han ido realizando. Por lo tanto, se puede decir que el sistema cumple con los objetivos marcados y cubre la funcionalidad básica requerida. De esta manera, el sistema proporciona al usuario un conjunto de funcionalidades esenciales que hacen que la aplicación sea realmente útil y de ayuda, para que exista un progreso en los estudios del usuario, y posteriormente una mejora en sus notas.

Sin embargo, en el ámbito del rendimiento de la aplicación podrían aplicarse mejoras, puesto que, se interactúa en la mayoría de módulos funcionales con la base de datos y éstos están continuamente cargándose. En cada una de las peticiones de información a la base de datos se produce un pequeño tiempo de espera a la llegada de la información. De esta manera se produce demora, también, a la hora de navegar entre distintas pantallas.

Capítulo 6

6. Gestión del proyecto

6.1 Planificación

En este apartado se detalla la planificación del proyecto que se ha desarrollado.

En primer lugar, se define la planificación inicial, realizada antes de comenzar el proyecto.

A continuación, se define la planificación final, es decir, el progreso real que ha seguido el proyecto.

Planificación inicial

La planificación inicial del proyecto es una estimación del tiempo de desarrollo del proyecto que se realizó antes de comenzarlo.

Las etapas que componen la planificación serán detalladas a continuación:

- Fase de análisis.
  - Análisis del proyecto: Se realiza un primer estudio del proyecto que incluye un primer enfoque del mismo y el establecimiento de los objetivos que se persiguen.
  - Estudio del estado del arte: Se lleva a cabo un análisis del entorno tecnológico actual relativo a los dispositivos móviles.
  - Profundización en Android: Se realiza un repaso de los conocimientos adquiridos previamente sobre la plataforma Android. Asimismo, se realiza un estudio más profundo acerca del desarrollo de aplicaciones para este sistema operativo.
- Desarrollo del proyecto.
  - Implementación de la aplicación: En esta etapa se realiza la implementación de las funcionalidades básicas de la aplicación.

  - Implementación del servicio web: En esta fase se implementa el servicio web y se añaden las funciones necesarias en la aplicación para realizar llamadas al servicio web.
  - Integración con la base de datos: Se prepara la base de datos y se integra al funcionamiento en conjunto con la aplicación y el servicio web.
- Pruebas.
  - Pruebas del sistema: En esta etapa se realizan las pruebas necesarias para comprobar el correcto funcionamiento del sistema.
- Documentación.
  - Redacción de la memoria: En esta etapa se redacta el presente documento con la información que se ha ido recopilando a lo largo del desarrollo del proyecto.

La Figura 62 refleja la planificación inicial del proyecto mediante un diagrama de Gantt.

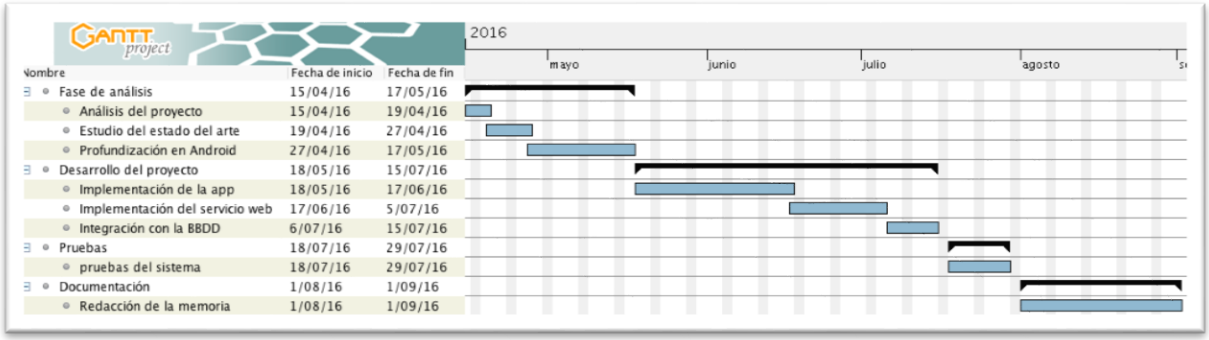

Figura 62 - Planificación inicial.

Planificación final

La planificación final detalla la evolución real del proyecto. No se ha podido cumplir con los períodos estimados en la planificación inicial.

La fase de análisis es la etapa que más se ha acercado a los plazos establecidos en la planificación inicial.

La etapa de desarrollo del sistema ha sido el que más tiempo ha ocupado. Comparándolo con la estimación de esta etapa en la planificación inicial ha sido la etapa que más retraso ha sufrido debido a las dificultades durante la implementación, la inclusión de nuevas funcionalidades y las numerosas revisiones realizadas.

La etapa de pruebas se ha ampliado, llegando a iniciarse antes debido a la necesidad de realizar pruebas durante el desarrollo del sistema.

La fase de redacción de la memoria también se ha extendido más allí de lo previsto en la planificación inicial, debido a una mala estimación de la duración de la redacción y las numerosas revisiones realizadas del documento. Asimismo, el período de redacción de la memoria ha comenzado antes de lo planificado inicialmente para que la documentación se lleve de manera paralela con el progreso del proyecto. De esta manera, el contenido que se va incluyendo está actualizado a medida que se avanza el proyecto y se garantiza que no se olvida incorporar contenido importante.

La Figura 63 muestra el diagrama de Gantt con el progreso real del proyecto.

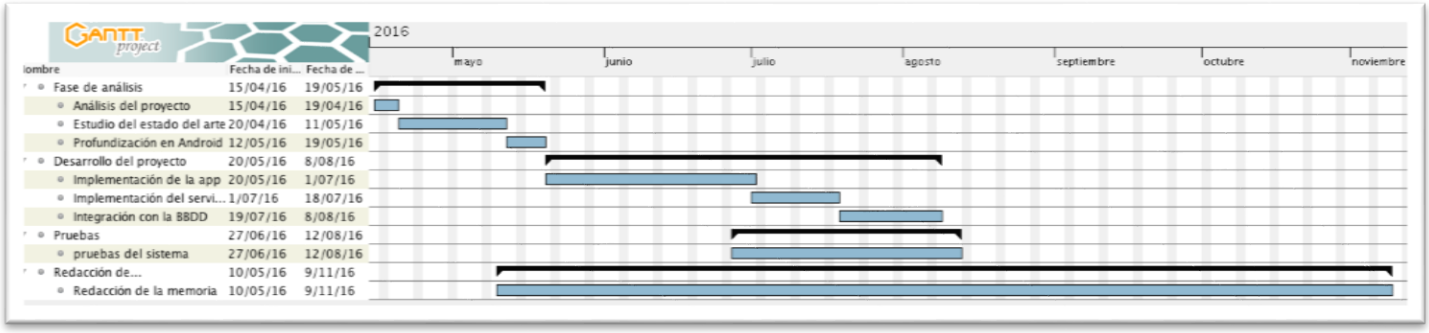

Figura 63 - Planificación final.

6.2 Presupuesto

En este apartado se detallan los costes producidos durante el desarrollo del proyecto. El presupuesto se desglosa en los costes asociados a las horas trabajadas por el personal y costes de los materiales empleados en la realización de este proyecto.

6.2.1 Costes de personal

El coste de personal recoge los gastos asociados a las horas trabajadas por el alumno que ha participado en el desarrollo del proyecto (ver Tabla 44). Se ha desglosado las horas trabajadas para cada una de las tareas realizadas para el desarrollo del proyecto.

Se ha estimado que el coste de un ingenierío informático recién graduado es de alrededor de 13 euros por hora trabajada.

Costes de personal		
Tarea realizada	Horas empleadas[horas]	Coste [€]
Fase de análisis	60	780,00 €
Fase de desarrollo	150	1950,00 €
Fase de pruebas	25	325,00 €
Documentación	70	910,00 €
Total	305	3965,00 €

Tabla 44. Costes de personal.

6.2.2 Costes de materiales

Este apartado concreta los costes de los materiales, tanto hardware como software, utilizados para elaborar el presente proyecto.

La Tabla 45 recoge los datos de los materiales hardware implicados en el desarrollo del sistema.

	Presupuesto					
	Material	Precio [€]	Amortización [meses]	Amortización [€/mes]	Período de uso [meses]	Coste del proyecto [€]
Hardware	Ordenador portátil	1350,00	60	22,50	8	180,00
	Smartphone Android	500,00	36	13,89	3	41,67
	Servidor de alojamiento (Hostinger)	0,00	12	0,00	2	0,00
Software	Android Studio IDE	0,00	-	-	4	0,00
	Licencia de desarrollador de Android	23,58	-	-	4	23,58
	Microsoft Office 365 Personal (Mac OS X)	69,00	12	5,75	6	34,50
	Filezilla	0,00	-	-	1	0,00
	Motor de síntesis de voz de Google	0,00	-	-	2	0,00
	Gantt Project	0,00	-	-	1	0,00
	Postman	0,00	-	-	1	0,00
	Volley	0,00	-	-	2	0,00
Total						279,75

Tabla 45. Costes materiales.

6.2.3 Resumen de costes

Para finalizar con el presupuesto, la tabla 46 contiene el resumen de costes totales del presente proyecto. El resumen de los costes recoge los costes de personal, los costes materiales y el coste total del proyecto.

Concepto	Coste [€]
Costes de personal	3965,00 €
Costes materiales	279,75€
Total	4244,75€

Tabla 46. Resumen de costes.

El coste total de este proyecto asciende a un total de CUATRO MIL DOSCIENTOS CUARENTA Y CUATRO EUROS con SETENTA Y CINCO CÉNTIMOS.

6.3 Entorno socioeconómico

Después de haber calculado los costes que ha supuesto el desarrollo de la aplicación, a lo largo del apartado 6.2, es necesario exponer la manera en la que se puede recuperar la inversión y generar beneficios.

La primera opción es publicar la aplicación en la plataforma Google Play, puesto que ya se posee la licencia de desarrollador de Android. La aplicación sería distribuida como descarga gratuita con un banner de publicidad, para generar beneficios, en la parte inferior de la pantalla. Además se daría la opción a los usuarios de utilizar la aplicación sin publicidad mediante un pago de 0,50€.

La segunda opción sería que la aplicación fuese gratuita y sin publicidad hasta conseguir un considerable número de usuarios registrados y buscar la venta de la aplicación a una empresa dedicada al sector o a una institución educativa.

Otra opción es la de publicar la aplicación en la plataforma Google Play como aplicación de pago con un precio de descarga de 0,50€. Sin embargo, no se tiene la intención de obligar al usuario a pagar para utilizar la aplicación. Por consiguiente, es la opción menos probable.

6.4 Marco regulatorio

La aplicación pertenece al campo del e-learning, más concretamente al m-learning. Por ello, se debe establecer el marco regulatorio que corresponde al proyecto.

Se tiene presente la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal [45], mediante la cual, se protegerán los datos y la información de los usuarios, utilizándolos únicamente en el contexto de la aplicación.

Se tiene en cuenta la Ley 34/2002, de 11 de julio, de Servicios de la Sociedad de la Información y de Comercio Electrónico [46], por la cual se regula la contratación de bienes y servicios por vía electrónica y el suministro de información por dicho medio siempre que represente una actividad económica para el prestador. Esto aplica en el caso de que se monetice la aplicación, ya sea mediante publicidad o poniendo precio a su descarga (apartado 6.3). Se informará debidamente a los usuarios acerca de los datos del responsable de la aplicación, datos de contacto, precio de los servicios que se ofrecen y se deberá obtener el consentimiento del usuario.

Se tiene presente la Ley 59/2003, de 19 de diciembre, de firma electrónica [47], que regula la firma electrónica, su eficacia jurídica y la prestación de servicios de certificación. Android exige que todos los APK se firmen digitalmente con un certificado para su instalación. Esto permite que Android pueda verificar que cualquier actualización futura de la aplicación sea auténtica y provenga del autor original.

Asimismo se tiene en cuenta el Real Decreto Legislativo 1/1996, de 12 de abril, de la Ley de Propiedad Intelectual [48], que incluye las normas relativas a la propiedad intelectual, los derechos de autor y condiciones para el uso de obras de dominio público. Por lo tanto, se tomarán las medidas necesarias para el uso de material o contenido de terceras personas en la aplicación. A pesar de que, de momento, la aplicación utiliza únicamente contenido propio, se tendrá presente esta normativa para hacerlo con su correspondiente autorización, en un futuro.

Capítulo 7

7. Conclusiones y trabajo futuro

En este capítulo se exponen las conclusiones generales y personales acerca del desarrollo del proyecto. Asimismo, se recogen las líneas de trabajo futuro, que plantean posibles mejoras sobre el sistema desarrollado y la posibilidad de incorporar nuevas funcionalidades, en la aplicación, no implementadas en este proyecto.

7.1 Conclusiones

Las conclusiones acerca del proyecto recogen los objetivos logrados y las impresiones personales durante su desarrollo.

En relación a los objetivos propuestos y los objetivos cumplidos, se ha alcanzado un grado alto de éxito. El principal objetivo era proporcionar una aplicación que sirva de ayuda, refuerzo y apoyo a los estudiantes universitarios a lo hora de preparar sus exámenes. Se ha desarrollado la aplicación siguiendo la descripción realizada en el apartado 1.2.

Asimismo, se ha llevado a cabo un estudio y la realización de los siguientes puntos:

- Profundizar los conocimientos de la plataforma Android y de su entorno de desarrollo de aplicaciones. (ver apartado 3.2).
- Realizar un análisis de los sistemas de diálogo hablado para desarrollar aplicaciones que proporcionen reconocimiento del habla (ver apartado 2.3).
- Realizar un análisis de las tecnologías y herramientas que se van a emplear a la hora de desarrollar la aplicación, valorando las distintas posibilidades para implementar una arquitectura cliente-servidor. (ver apartado 3.3 y apartado 3.4).
- Presentar ejemplos de aplicaciones alternativas ya existentes que sean similares a la aplicación que se va a desarrollar en este proyecto. (ver apartado 2.4).

De la misma manera, se ha utilizado tecnologías con las que se tenía poca práctica o algunas con las que no se había trabajado previamente. De ellas hay que comentar que ha sido muy útil profundizar el conocimiento sobre ellas o aprender a utilizarlas, tanto para el desarrollo de este proyecto como para proyectos futuros. Entre ellas destacan, los servicios web, utilizados en un par de asignaturas de la carrera, pero no en gran profundidad; PHP, un lenguaje muy sencillo para programar la parte del servidor, que se ha tenido que aprender para realizar este proyecto y la librería *Volley* de Android que permite realizar conexiones a través de internet de una manera sencilla y cómoda.

En relación a las impresiones personales, este trabajo ha supuesto un gran reto personal y profesional. Se ha dedicado muchas horas de trabajo a la realización del proyecto, para recopilar información e investigación, desarrollo de la aplicación, redacción de la memoria, etc. Además, la realización de este trabajo ha dado un nuevo enfoque a los conocimientos adquiridos durante la carrera y ha abierto un nuevo camino hacia donde encaminar el futuro laboral.

Por último, se puede afirmar que la aplicación ha alcanzado los objetivos principales marcados y que el grado de satisfacción tras la realización del proyecto es bastante alto.

7.2 Trabajos futuros

En este apartado se definen las líneas de trabajo futuro para el proyecto desarrollado a lo largo de este trabajo de fin de grado. Estas líneas consideran las posibles mejoras encontradas para proporcionar continuidad al proyecto y añadir nuevas funcionalidades a la aplicación una vez se haya concluido el trabajo de fin de grado. A continuación, se exponen las mejoras propuestas:

- La primera mejora propuesta es el multilenguaje, es decir, se añadirían nuevos idiomas a modo de alternativa al español como lenguaje de la interfaz, así como del contenido de la aplicación para abarcar más mercado y tener la posibilidad de llegar a más usuarios.
- Incorporar la posibilidad a los usuarios de acceder a la aplicación utilizando sus cuentas en redes sociales como Facebook, Twitter y Google. Esta funcionalidad se puede implementar utilizando las APIs proporcionadas por las propias compañías.

- Añadir feedback auditivo, mediante sonidos o tonos, que complemente al feedback visual ya implementado. Para, de esta manera, mejorar la experiencia del usuario con la aplicación.
- Implementar un mecanismo de "escucha continua" de manera que la aplicación realice el reconocimiento de voz sin necesidad de pulsar el botón destinado a ello.
- Incluir contenido, temario y preguntas de otras carreras universitarias, o campos de estudio puesto que de momento hay únicamente contenido del grado universitario que se ha cursado. Para realizar este punto es necesario consultar a expertos en dichos campos o carreras para garantizar que se ofrece un contenido completo y correcto.
- Aumentar la cantidad de preguntas que hay actualmente alojadas en la base de datos, de manera que éstas sean más variadas y abarquen más contenido.
- Añadir una sección tutorial que sirva a modo de explicación de uso de la aplicación, enseñando las distintas maneras de interactuar con ésta, utilizando elementos gráficos como texto, iconos, animaciones o vídeos que hagan más sencillo el aprendizaje.
- Incluir un sistema de valoración de las preguntas que se realizan durante los test en la que los usuarios puntúan las preguntas que van contestando en función de la calidad o exactitud de las mismas.
- Mejorar la seguridad del sistema considerando que no se han tomado algunas medidas de seguridad básicas, en esta primera versión para proteger el sistema de inyecciones SQL, ofuscar el código o cifrar toda la información que sea almacenada en la base de datos.
- Añadir medidas de acceso a la aplicación que garanticen a los usuarios que terceras personas no puedan entrar en sus cuentas y sus datos mediante el bloqueo de acceso tras cinco intentos y la limitación de acceso a una cuenta desde un único dispositivo a la vez.
- Elaborar una base de datos local que almacene los datos necesarios para que el usuario pueda utilizar la aplicación aún cuando no tenga acceso a internet.

Glosario

- **Android:** Sistema operativo móvil desarrollado por Google, basado en el kernel de Linux y diseñado para dispositivos móviles con pantalla táctil.
- **API (*Application Programming Interface*)**: Un conjunto de funciones, subrutinas y métodos de una librería que son utilizados por otro software.
- **APK (*Android Application Package*)**: Es un formato de empaquetado que utiliza el sistema operativo Android para la distribución e instalación de aplicaciones móviles.
- **Aplicación:** Programa informático diseñado que permiten al usuario realizar un conjunto de tareas, funciones o actividades coordinadas.
- **Bluetooth:** Estándar de tecnología inalámbrica para el intercambio de datos de dispositivos fijos y móviles en distancias cortas.
- **E-learning:** Metodología de aprendizaje mediante la tecnología.
- **Framework:** Esquema, esqueleto o patrón que se utiliza para el desarrollo y la implementación de una aplicación.
- **Feedback:** Mecanismo por el cual la señal de salida de un sistema se redirige al usuario con la finalidad de controlar su comportamiento.
- **Hardware:** Conjunto de componentes físicos o materiales que constituyen un computador o un sistema informático.
- **Hosting:** Servicio que se proporciona a los usuarios de Internet para poder almacenar cualquier contenido accesible vía web.
- **Interfaz:** Conexión funcional entre dos sistemas que proporciona una comunación de distintos niveles para el intercambio de información.
- **Librería:** Conjunto de recursos utilizados para desarrollar software.
- **Middleware:**
- **M-learning:** Metodología de enseñanza y aprendizaje mediante la utilización de dispositivos móviles.
- **Mockup:** Diseño de una interfaz de usuario hecho en papel o hecho con imágenes en ordenador.
- **Multi-touch:** Tecnología que permite que una pantalla táctil reconozca la presencia de más de un punto de contacto con su superficie.

- **NFC (*Near Field Communication*):** Protocolo de comunicación que permite el intercambio de datos entre dispositivos en distancias cortas.
- **Ofuscar:** Cambio en el código fuente de un programa informática con la finalidad de que sea difícil de entender o leer.
- **Open source:** Software cuyo código fuente publicado bajo una licencia que permite a los usuarios cambiar o mejorar el software y redistribuirlo, ya sea en su forma original o modificada.
- **Smartphone:** Teléfono móvil con un sistema operativo, optimizado para dispositivos móviles, que cuenta con una gran capacidad de almacenamiento y muy buena conectividad.
- **Software:** Conjunto de componentes lógicos que constituyen un sistema informático.
- **Tablet:** Computador móvil plano y de tamaño reducido, ligeramente mayor que un smartphone, que cuenta con una pantalla táctil y batería recargable.
- **Tethering (anclaje a red):** Proceso por el cual un dispositivo móvil con conexión a Internet actúa como pasarela para ofrecer acceso a la red a otro dispositivo.
- **VoIP:** Conjunto de recursos que permiten la comunicación por voz a través de Internet empleando el protocolo IP.
- **Widget:** Elemento gráfico de interacción que es parte de una interfaz de usuario.
- **Wi-Fi hotspot (Smartphone):** Proceso por el cual un dispositivo móvil con conexión a Internet proporciona acceso a la red a otro dispositivo mediante conexión Wi-Fi.

Bibliografía

[1] Fernando Rivero, “Informe ditrendia 2016: Mobile en España y en el mundo”. [Online]. Disponible: http://www.ditrendia.es/wp-content/uploads/2016/07/Ditrendia-Informe-Mobile-en-España-y-en-el-Mundo-2016-1.pdf

[2] El Mundo, “España sigue liderando el ránking de abandono escolar en la UE”. Abril de 2016. [Online]. Disponible: http://www.elmundo.es/sociedad/2016/04/27/5720a75622601d78798b466e.html

[3] César Tardáguila Moro, “Dispositivos móviles y multidia”. [Online]. Disponible: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/9164/1/dispositivos_moviles_y_multimedia.pdf

[4] N-economía, “Uso del Smartphone y Tablet en España”. Enero de 2016. [Online]. Disponible: http://n-economia.com/notasalerta/uso-del-smartphone-y-la-tablet-en-espana-infografia/

[5] Samuel Fernández, “España, territorio smartphone”. Enero de 2016. [Online]. Disponible: http://www.xatakamovil.com/movil-y-sociedad/espana-territorio-smartphone

[6] Gartner, “Gartner Says Worldwide Smartphone Sales Grew 9.7 Percent in Fourth Quarter of 2015”. Febrero de 2016. [Online]. Disponible: http://www.gartner.com/newsroom/id/3215217

[7] Juan Otálora Alarcón “Android Wear, Tv y Auto: El futuro de las tecnologías”. [Online]. Disponible: https://rincondelatecnologia.com/android-wear-tv-y-auto-el-futuro-de-las-nuevas-tecnologias/

[8] Pablo Espeso, “De Cupcake a Marshmallow”. Agosto de 2015. [Online]. Disponible: http://www.xatakamovil.com/sistemas-operativos/de-cupcake-a-marshmallow-asi-han-sido-las-versiones-de-android-a-lo-largo-de-su-historia

[9] Social compare, “Android versions comparison” [Online]. Disponible: http://socialcompare.com/en/comparison/android-versions-comparison

[10] Wired, "Android 2.2 'Froyo' features USB, Wi-Fi tethering'" [Online]. Disponible: https://www.wired.com/2010/05/android-22-froyo-features-usb-wi-fi-tethering/

[11] Parabal, "Android vs iOS mobile operating systems". [Online]. Disponible: http://www.parabal.com/whitepapers/iOS_vs_Android.pdf

[12] Verge Staff, "iOS: A visual history". [Online]. Disponible: http://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad

[13] Techradar, "iOS 10 release date, news and features". Septiembre de 2016. [Online]. Disponible: http://www.techradar.com/news/software/operating-systems/ios-10-release-date-news-beta-and-rumors-1311275

[14] Neowin, "From Photon to the future. A not-so-brief history of Windows Phone". Febrero de 2016. [Online]. Disponible: https://www.neowin.net/news/from-photon-to-the-future-a-not-so-brief-history-of-windows-phone

[15] IDC, "Worldwide Smartphone Growth Forecast to Slow to 3.1% in 2016 as Focus Shifts to Device Lifecycles, According to IDC". Junio de 2016. [Online]. Disponible: http://www.idc.com/getdoc.jsp?containerId=prUS41425416

[16] Android-developers, "Android Market. Now available for users". [Online]. Disponible: http://android-developers.blogspot.com.es/2008/10/android-market-now-available-for-users.html

[17] Apple, "Apple Developer Program". [Online]. Disponible: https://developer.apple.com/programs/

[18] Microsoft, "Account types, location and fees". [Online]. Disponible: https://msdn.microsoft.com/en-us/windows/uwp/publish/account-types-locations-and-fees

[19] R. López-Cortázar, "Sistemas de diálogo hablado y multimodal". [Online]. Disponible: http://www.ugr.es/~rlopezc/sistemas_dialogo.htm

[20] J. Llisterri, "Introducción a los sistemas de diálogo" [Online]. Disponible: http://liceu.uab.cat/~joaquim/publicacions/Llisterri_06_Sistemas_Dialogo.pdf

[21] D. Griol Barres, "Desarrollo y evaluación de diferentes metodologías para la gestión automática de diálogo". [Online]. Disponible: http://users.dsic.upv.es/~dgriol/papers/TesisDgriol.pdf

[22] Andrew Sutherland, "Meet the new Quizlet". [Online]. Disponible: https://quizlet.com/blog/meet-the-new-quizlet

[23] Duolingo, "Aprende un idioma gratis y diviértete". [Online]. Disponible: https://es.duolingo.com

[24] Android, "The Android source code". [Online]. Disponible: https://source.android.com/source/index.html

[25] Tomás Girones, J., (2013), *El gran libro de Android*, Barcelona, España, Marcombo.

[26] Android, "Platform architecture". [Online]. Disponible: https://developer.android.com/guide/platform/index.html

[27] Komatineni, S., MacLean, D., (2012), *Pro Android 4. Android 4 Platform SDK technniques for developing smartphone and tablet apps*, Apress.

[28] Cancela García, L., Ostos Lobo, S., "Aplicaciones en Android". [Online]. Disponible: https://sites.google.com/site/swcuc3m/home/android/generalidades/aplicacionesandroid

[29] Android, "App Components: Activity", [Online]. Disponible: https://developer.android.com/guide/components/activities.html

[30] Oculus , "Android Development Software Setup for Mac OS X". [Online]. Disponible: https://developer3.oculus.com/documentation/mobilesdk/latest/concepts/mobile-studio-setup-android-mac/

[31] Clark, Bryan, "How to set up Android Studio on your Mac". [Online]. Disponible: http://www.makeuseof.com/tag/set-android-studio-mac-youd-want/

[32] Android Developers Blog "An update on Eclipse Android Developer Tools". [Online]. Disponible: http://android-developers.blogspot.de/2015/06/an-update-on-eclipse-android-developer.html

[33] Berners-Lee, T., Fielding, R., Frystyk, H., "HyperText Transfer Protocol". [Online]. Disponible: https://tools.ietf.org/html/rfc1945

[34] Belmonte Fernández, O. "Introducción al lenguaje de programación Java. Una guía básica". [Online]. Disponible en: http://www.academia.edu/6336525/Introducción_al_lenguaje_de_programación_Java._Una_guía_básica

[35] PHP, "¿Qué es PHP?". [Online]. Disponible en: http://php.net/manual/es/intro-whatis.php

[36] W3C, "Guía breve de tecnologías XML". [Online]. Disponible en: http://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML

[37] Hostinger, "Hosting web". [Online]. Disponible en: http://www.hostinger.es/hosting-web

[38] Sammarti, S. "Introducción a MySQL". [Online]. Disponible en: http://www.esandra.com/mysql-i-introduccion-a-las-bases-de-datos-relacionales/

[39] PhpMyAdmin, "phpMyAdmin bringing MySQL to the web". [Online]. Disponible en: https://www.phpmyadmin.net

[40] Álvarez, C., "Chrome REST y Postman". [Online]. Disponible en: http://www.genbetadev.com/herramientas/chrome-rest-y-postman

[41] OBS Business School, "Gantt Project: Análisis del software". Disponible en: http://www.obs-edu.com/es/blog-project-management/diagramas-de-gantt/ganttproject-analisis-del-software

[42] Android, "Transmitting network data using Volley". Disponible en: https://developer.android.com/training/volley/index.html

[43] Fernández, A., "Servicios web RESTful con HTTP. Introducción y bases teóricas". Disponible en: http://www.adwe.es/general/colaboraciones/servicios-web-restful-con-http-parte-i-introduccion-y-bases-teoricas

[44] Bartolomé, R., "A simple web service using PHP, JSON and MySQL". Disponible en: http://robertobartolome.com/a-simple-web-service-using-php-json-and-mysql/

[45] “Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal”. Agencia Estatal Boletín Oficial del Estado. Disponible en: https://www.boe.es/buscar/act.php?id=BOE-A-1999-23750

[46] “Ley 34/2002, de 11 de julio, de Servicios de la Sociedad de la Información y de Comercio Electrónico”. Agencia Estatal Boletín Oficial del Estado. Disponible: https://www.boe.es/buscar/act.php?id=BOE-A-2002-13758

[47] “Ley 59/2003, de 19 de diciembre, de firma electrónica”. Agencia Estatal Boletín Oficial del Estado. Disponible en: https://www.boe.es/buscar/doc.php?id=BOE-A-2003-23399

[48] “Real Decreto Legislativo 1/1996, de 12 de abril, de la Ley de Propiedad Intelectual”. Agencia Estatal Boletín Oficial del Estado. Disponible en: https://www.boe.es/diario_boe/txt.php?id=BOE-A-1996-8930